Università di Pisa
Facoltà di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Informatica
Corso di Laurea Specialistica in Informatica

Tesi di laurea

Stima dell'illuminazione di una scena tramite rendering inverso

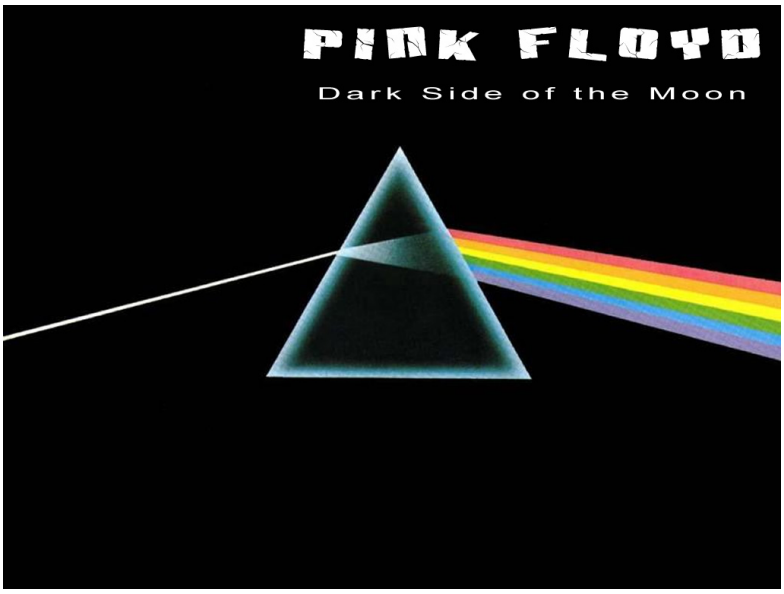
Candidato:
Mario Latronico

Relatori:
Dott. Paolo Cignoni

Ing. Massimiliano Corsini

Controrelatore:
Prof. Roberto Grossi

Anno Accademico 2005/2006



Pink Floyd, The Darkside of the Moon

Alla mia famiglia,
per il supporto datomi durante questi anni di studio.

Ad Elena,
che mi sopporta e mi infonde coraggio in ogni situazione.

Stima dell'ambiente di illuminazione tramite rendering inverso

Mario Latronico

Università di Pisa
Dipartimento di Informatica

Sommario

Spesso è utile, dopo aver scattato una fotografia, intervenire su alcune sue caratteristiche, ad esempio per modificare le ombre portate dagli oggetti oppure per cambiare il materiale o il colore delle superfici. La tesi propone una tecnica che consente di ricavare l'ambiente di illuminazione di una scena a partire da una fotografia e dai modelli geometrici tridimensionali della scena stessa.

Indice

Ringraziamenti

Desidero ringraziare prima di tutto i miei relatori, Paolo Cignoni e Massimiliano Corsini: senza il loro aiuto e i loro consigli sarei ancora su queste righe.

Ringrazio inoltre tutti i ricercatori, i dottorandi e i colleghi tesisti del Visual Computing Group del CNR di Pisa: Matteo Dellepiane, per aver sviluppato il programma TexAlign, fondamentale nello svolgimento di questa tesi e per la sua simpatia; Marco Di Benedetto, Valentino Fiorin, Guido Ranzuglia e Andrea Venturi per aver rallegrato le mie giornate passate nel laboratorio; Marco Callieri e Marco Tarini per i consigli utili nella risoluzione di alcuni problemi; Giuseppe Croccia, Francesca De Mitry, Fabio Ganovelli, Federico Ponchio e Andrea Spinelli.

Ringrazio tutti coloro che in questi anni mi hanno sopportato, consolato e rallegrato: gli amici "della vacanza a Tropea" Alan Pietrini, Riccardo Delnevo, Susanna Tian, Eugenio Pulsinelli, Cristian Cecina, Manolo Garabini, Chiara, Federico Lombardi, Susanna Bertucci, Riccardo Cecchini, Giacomo Giangare, Serena Brancato e Davide Donaggio; gli amici di vecchia data Marcello Trafossi, Damiano di Mauro, Tommaso Ferrari e Giacomo Tocci; i colleghi e amici dell'università Michele Girolami, Andrea Bresciani, Martina Galletto, Marco Luisi, Francesco Monaci, Matteo Demuru e Massimiliano Conti.

Infine un doppio ringraziamento è rivolto a chi mi ha aiutato ma ho

dimenticato di citare.

Capitolo 1

Introduzione

Fin dagli albori della computer grafica la produzione di immagini fotorealistiche è stata uno degli obiettivi principali di studiosi ed artisti. Il processo di *rendering* genera un'immagine a partire dalla descrizione di una scena, la quale è costituita da uno o più modelli tridimensionali (generalmente specificati in maniera discreta come insieme di poligoni connessi fra loro), dai materiali che li caratterizzano, da un'insieme di luci che illuminano la scena e da una camera che definisce l'inquadratura. Durante gli anni sono stati sviluppati modelli matematici e algoritmi, come il *radiosity* e il *raytracing*, che consentono di ottenere immagini sintetiche praticamente indistinguibili da fotografie reali (come è possibile vedere in figura 1.1), simulando in maniera coerente l'interazione della luce con la materia, tanto che al giorno d'oggi il realismo è spesso limitato soltanto dalla risoluzione dei modelli geometrici tridimensionali. Entrambe le tecniche sopra menzionate consentono di simulare la cosiddetta *illuminazione globale*, la quale tiene conto non soltanto della luce che arriva direttamente da una sorgente luminosa, ma anche dell'*illuminazione indiretta*, che proviene dalla riflessione della luce sulle altre superfici; infatti quando un raggio luminoso colpisce una superficie questa assorbirà parte dell'energia luminosa, mentre la restante verrà riflessa.

Il radiosity simula questo comportamento dividendo le superfici degli oggetti in *patch*, cioè elementi geometrici di base planare. Ogni patch, di

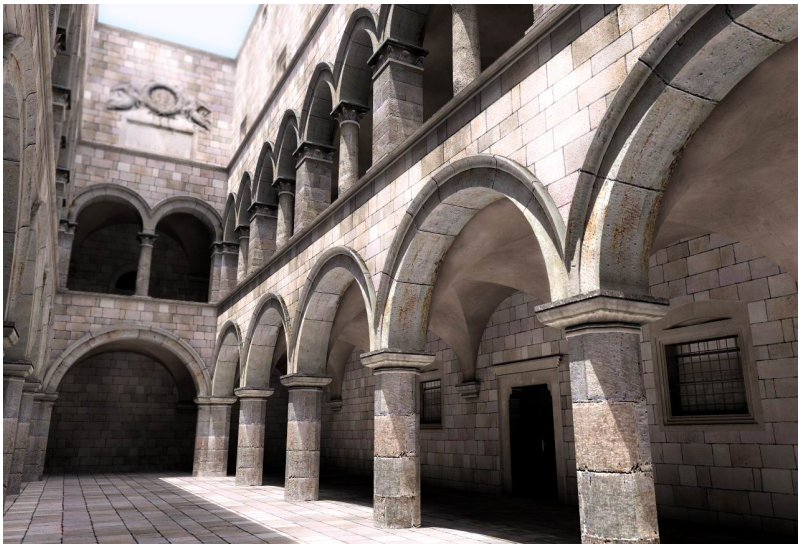


Figura 1.1: *Sponza Atrium, render di Gianni Melis, modello RNA studios.*

grandezza finita, assorbe una certa quantità di luce e ne riflette la restante.

Tramite il ray tracing invece è possibile simulare in maniera realistica il fenomeno della riflessione e della rifrazione. Esso funziona tracciando il percorso che può intraprendere un ipotetico raggio di luce che interseca la lente della camera. In realtà, poichè la maggior parte dei raggi che hanno origine da una sorgente di luce non intersecheranno la lente della camera, per motivi di efficienza viene simulato il percorso inverso, ossia i raggi partono dalla camera fino ad arrivare ad una sorgente di luce oppure si perdono all'infinito. Se il raggio durante il suo percorso incontra un'oggetto trasparente, questo viene deviato a seconda dell'indice di rifrazione dell'oggetto. Similmente, se l'oggetto è riflettente (ad esempio un cofano di un'automobile), il raggio devia secondo la legge fisica della riflessione, per la quale l'angolo di incidenza è uguale all'angolo di riflessione. Se l'oggetto è opaco e non riflettente verrà restituito il pixel corrispondente al punto di intersezione del raggio con la superficie dell'oggetto. Infine se nessuno di questi casi è verificato il valore

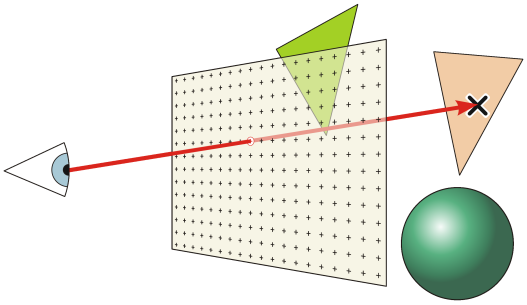


Figura 1.2: *Schema del funzionamento del raytracing*

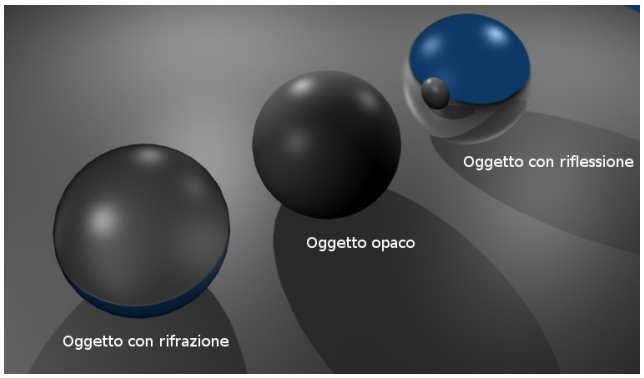


Figura 1.3: *Tre oggetti che illustrano la rifrazione, la riflessione e l'opacità*

del pixel sarà quello dello sfondo della scena.

Tracciando una grande quantità di raggi è possibile determinare i valori dei pixel che compongono l'immagine. In figura 1.3 è possibile vedere un esempio di immagine calcolata con raytracing che illustra queste proprietà.

Grazie al progresso nell'hardware e del software in campo grafico, è oggi possibile disporre su normali personal computer di una notevole potenza di calcolo che consente di ottenere in tempo reale immagini sempre più vicine alla realtà. L'uso di immagini generate al computer sinteticamente si è quindi diffuso nei più disparati campi: dall'intrattenimanto, ad esempio nel cinema e nei videogames, alla ricerca scientifica, ad esempio nei software di

visualizzazione tridimensionale delle proteine o nei software di telemedicina, fino ai musei virtuali.

Tuttavia a volte è necessario intervenire su una fotografia per poterla modificare *dopo* che questa è stata scattata, ad esempio per cambiare le condizioni di illuminazione o il colore degli oggetti che appaiono in una scena di un film. Esistono da parecchi anni dei software di fotoritocco bidimensionali che consentono di migliorare la qualità o di intervenire artisticamente su immagini statiche o in movimento, ad esempio per migliorare il contrasto di un'immagine oppure per filtrarla dal rumore eccessivo introdotto dai CCD delle fotocamere digitali. Purtroppo questi software agiscono su input bidimensionali (la fotografia è vista come matrice di pixel) e restituiscono immagini bidimensionali. Alcuni compiti, come la rimozione delle ombre e il cambiamento delle condizioni di luce risultano ancora di difficile applicazione. In questo caso si possono sfruttare gli algoritmi di rendering, poichè essi sono pensati per simulare proprio il comportamento fisico della luce, applicandoli in maniera inversa. Avendo ad esempio a disposizione una fotografia della scena, una descrizione della geometria degli oggetti e una descrizione dei loro materiali è possibile determinare le condizioni di illuminazione al momento dello scatto della foto.

Infatti consideriamo un versione semplificata *dell'equazione di rendering* ([12], che verrà spiegata in maniera più completa nel capitolo seguente):

$$I = \mathbf{M}(d + \mathbf{G}I) \tag{1.1}$$

I è la luce emessa da ogni punto della scena, $\mathbf{M}$ è il fattore che indica come le superfici riflettono la luce e d è l'illuminazione diretta dalle luci, mentre il restante termine indica l'illuminazione indiretta, dove $\mathbf{G}$ è la geometria e il secondo termine I è la luce che viene riflessa dalle superfici (l'equazione è ricorsiva). Il fattore M viene anche chiamato BRDF (*Bidirectional Reflectance Diffuse Function*) e verrà spiegato più estesamente nel capitolo successivo.

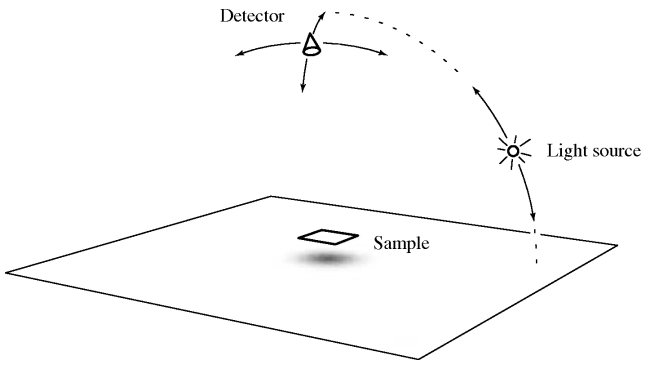


Figura 1.4: *Schema di gonioreflettometro [13].*

E' possibile ricavare un termine di quest'equazione essendo a conoscenza degli altri. Nel nostro caso siamo interessati a d per poter stimare l'ambiente di illuminazione della foto, I, scattata in precedenza. Il termine $\mathbf{G}I$ descrive come la luce interagisce tra una superficie e un'altra.

Tuttavia spesso non si hanno a disposizione tutti i termini ed è quindi necessario introdurre delle ipotesi che conducano ad una soluzione univoca; in molti casi non è possbile determinare M perchè si ha soltanto una foto di un oggetto di cui si dispone anche della geometria G acquisita con uno scanner tridimensionale. Molte superfici, ad esempio la creta, possono essere assimilate a superfici cosiddette *Lambertiane*, per le quali la luce viene riflessa in modo uniforme ed è quindi indipendente dal punto di vista dell'osservatore, ossia non ci sono riflessi speculari (approfonditi in 2.1.2) che cambierebbero con il punto di vista dell'osservatore come nei materiali plastici. In questo caso M diventa noto ed è più facile la stima dell'illuminazione. Diversamente per determinarlo sarebbe necessario un *gonioreflettometro*, ossia un dispositivo che ha una sorgente di luce la quale illumina il materiale ed un sensore che cattura la luce *riflessa* dal materiale. Attraverso la rotazione sia del sensore che della luce è possibile stimare i parametri del materiale. Uno schema di tale dispositivo è mostrato in figura 1.4.

A volte invece potremmo essere interessati al valore di M, avendo a disposizione gli altri termini come parametri. Tuttavia, M è una funzione complessa di 4 parametri (il gonioreflettometro ha 4 gradi di libertà, due di rotazione per il sensore e altrettanti per la luce) e per poterla stimare accuratamente senza l'uso di questo dispositivo è necessario disporre di più fotografie e della geometria.

1.1 Obiettivi e struttura della tesi

L'obiettivo di questa tesi è la stima di un'ambiente di illuminazione a partire da una geometria nota, precedentemente acquisita tramite scanner tridimensionali, e da una BRDF stimata in maniera manualmente in maniera iterativa. Il lavoro svolto si inserisce nel contesto della visualizzazione nel settore dei Beni Culturali che è un' area di interesse del Visual Computing Group del CNR di Pisa. Infatti, per poter avere una rappresentazione digitale realistica di un manufatto reale, è necessario cambiare l'ambiente di illuminazione in modo da adattarlo a tutte le condizioni di visualizzazione possibili e non solo quella in cui è stato acquisito. Inoltre, avendo a dispozione l'ambiente di illuminazione, è possibile ad esempio inserire in maniera realistica un oggetto sintetico in una fotografia scattata in precedenza, oppure avere una stima migliore del colore di un oggetto.

La principale fonte di ispirazione per la tesi è stato il lavoro di Marschner [13]. Tuttavia, come nota lo stesso autore, la stima di un'ambiente di illuminazione è un problema mal condizionato; egli cerca di porvi rimedio introducendo un *coefficiente di correzione* il cui valore è impostato dall'utente; in questo lavoro abbiamo cercato di evitare il malcondizionamento tramite un approccio basato su *wavelet* (che è una rappresentazione di un segnale che verrà spiegata nel capitolo 3). Nel capitolo 2 si parlerà del rendering inverso e di come questo problema sia stato affrontato in precedenza e si introdurran-

no i concetti e la terminologia di base. Nel capitolo 3 verrà fatta una breve introduzione sulle wavelet e verrà spiegato il framework che è risultato dal lavoro svolto. Nel capitolo 4 si presenteranno invece i risultati ottenuti, confrontando i rendering con l'ambiente di illuminazione stimato dal framework rispetto alle fotografie reali. Infine nell'ultimo capitolo verranno presentate le conclusioni e alcune idee per gli sviluppi futuri.

Capitolo 2

Rendering Inverso

Il rendering inverso è un metodo per stimare i parametri di una scena reale, come l'ambiente di illuminazione o le texture, partendo da immagini. Diversi metodi sono stati presentati in precedenza, ad esempio da Marschner [13], Sato e al. [21], Devebec e al. [27], [24] e Ramamoorthi e al. [14], [20].

Il lavoro qui discusso si concentra in particolare sul problema dell'*inverse lighting*, che consente di ottenere la stima dell'ambiente di illuminazione di una scena avendo come parametri noti la geometria, la bidirectional reflectance diffuse function (BRDF) e la fotografia dell'ambiente reale. Prima di poter parlare dei metodi di stima è però necessaria una breve introduzione sul comportamento fisico della luce.

2.1 La fisica della luce

Tutti i software di rendering fotorealistici sono basati su un'approssimazione dell'equazione di rendering, presentata per la prima volta da Kajiya([12]), che descrive il flusso di energia luminosa in una scena tridimensionale

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) d\vec{w}' \qquad (2.1)$$

la quale esprime la luce uscente da una posizione x verso una direzione $\vec{w}$ come la luce emessa L_e dalla stessa posizione e verso la stessa direzione

più una quantità che dipende dal materiale ($f_r(x, \vec{w}', \vec{w})$, la *BRDF*, spiegata successivamente), dalla luce (L_i) che arriva in x da $\vec{w}'$ attenuata dall'angolo di incidenza ($\vec{w}' \cdot \vec{n}$). Questi termini devono essere integrati sulla semisfera (Ω) di tutte le direzioni entranti. Nel caso di materiali che non emettono la luce il termine L_e vale 0.

Il rendering e il suo inverso riguardano quindi la propagazione della luce nello spazio (descritta dalla *radiometria*). In questo lavoro vengono assunte due proprietà fondamentali della luce:

- la luce viaggia solo su linee rette
- gli effetti della luce sono lineari, ossia se più sorgenti di luce illuminano un oggetto, il risultato è equivante alla somma dei risultati ottenuti illuminando l'oggetto separatamente con ognuna delle sorgenti.

In radiometria si indica con *radianza* il flusso di luce attraverso lo spazio. Essa è definita come una funzione

$$L : \mathbb{R}^3 \times S^2 \to \mathbb{R}$$

su tutti i punti e su tutte le direzioni nello spazio[1] e misura il flusso dP di energia luminosa che incontra una superficie $d\mathbf{x}$ situata in $\mathbf{x}$ che guarda verso la direzione ω, contando solo un piccolo range di direzioni $d\omega$ intorno ad ω, in proporzione all'area di $d\mathbf{x}$ e l'angolo solido[2] $d\omega$. In formule

$$L(\mathbf{x}, \omega) = \frac{dP}{dx d\omega} \tag{2.2}$$

Essendo un flusso ha come unità di misura i Watt per metro quadro per steradianti. In realtà la radianza descrive il flusso attraverso la superficie quando questa è *perpendicolare* alla direzione di propagazione. Il flusso

[1] Con S^2 si indica l'insieme di tutte le direzioni nello spazio tridimensionale

[2] L'*angolo solido* corrisponde all'idea naturale di un'area di superficie sferica

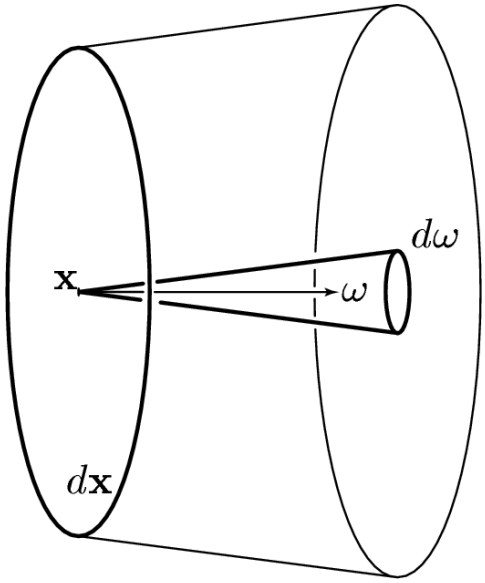


Figura 2.1: *Un elemento della superficie* $d\mathbf{x}$ *e un angolo solido* $d\omega$*.*

per unità d'area e per angolo solido attraverso una superficie orientata (non perpendicolare alla direzione di propagazione) con normale $\mathbf{x}$ è

$$L(\mathbf{x}, \omega) = \frac{dP}{dx d\omega \cos\theta} \tag{2.3}$$

dove θ è l'angolo tra la direzione ω e la normale $\mathbf{x}$.

L'*irradianza* è invece una misura dell'energia totale che cade su una superficie per unità d'area:

$$E(x) = \frac{dP}{dx} \tag{2.4}$$

mentre la *radianza uscente* è la misura dell'energia che esce da una superficie per unità d'area.

2.1.1 Bidirectional Reflectance Diffuse Function (BRDF)

In generale, quando la luce interagisce con la materia, avvengono tre tipi di interazione: la riflessione, l'assorbimento o la trasmissione della luce; una parte della luce incidente è riflessa, un'altra parte viene trasmessa (in dipendenza con il grado di opacità del materiale) e la restante viene assorbita dal materiale. La luce è una radiazione elettromagnetica e quindi per la legge di conservazione dell'energia vale la seguente equazione

$$L_{inc} = L_{refl} + L_{abs} + L_{trans} \tag{2.5}$$

dove L_{inc} è la luce incidente sulla superficie, che è la somma di quella riflessa, L_{refl}, di quella assorbita L_{abs} e di quella trasmessa L_{trans}. La *BRDF* (Bidirectional Reflectance Diffuse Function) è una funzione che descrive le caratteristiche del materiale di una superficie, indicando quanta luce viene riflessa quando essa incide sulla superficie. Essa dipende dalla direzione della luce entrante e del punto di vista dell'osservatore; ad esempio, guardando un oggetto di plastica illuminato da una luce bianca, si possono notare dei punti di specularità, che varieranno di posizione se l'osservatore si muove. Allo stesso modo, se l'oggetto e l'osservatore rimangono fermi e viene spostata la luce, cambierà la posizione dei punti di specularità.

La BRDF è quindi una funzione $f_r(\theta_i, \phi_i, \theta_e, \phi_e)$ data dal rapporto della radianza uscente osservata nella direzione $\omega_e = (\theta_e, \phi_e)$ con l'irradianza da un infinitesimo angolo solido attorno a $\omega_i = (\theta_e, \phi_e)$, dove ω_i, ω_e sono rispettivamente una direzione incidente e una direzione uscente, espresse in coordinate polari:

$$f_r(\omega_i, \omega_e) = \frac{L_o(\omega_e)}{E_i(\omega_i)} \tag{2.6}$$

dove L_o è la radianza uscente, cioè il flusso riflesso di energia per unità d'area per angolo solido e E_i è l'irradianza, cioè il flusso incidente per unità

d'area. Le principali proprietà della BRDF sono la simmetria rispetto ai suoi argomenti (detto anche *principio di reciprocità*):

$$f_r(\omega_i, \omega_e) = f_r(\omega_e, \omega_i) \tag{2.7}$$

e il rispetto della legge di conservazione dell'energia, la quale impone che quantità di luce uscente è sempre minore della luce entrante. Inoltre una BRDF può essere *isotropica* o *anisotropica*. Il termine isotropico è usato per descrivere una BRDF che ha proprietà di riflessione invarianti rispetto alla rotazione della superficie attorno alla sua normale; un esempio di tale comportamento è tipico dei materiali plastici. Le BRDF anisotropiche invece hanno proprietà di riflessione diverse rispetto ad una rotazione della superficie attorno alla sua normale; molti materiali metallici sono di questo tipo.

La BRDF è fondamentale nel processo di rendering per simulare la riflessione della luce e per ottenere immagini di sintesi fotorealistiche. Essa si può anche vedere come un operatore lineare $\mathbf{F_r}$ che mappa la distribuzione della radianza incidente nella distribuzione della radianza uscente, cioè $L_o = \mathbf{F_r} L_i$.

2.1.2 Fenomeni comuni di riflessione

Alcuni fenomeni di trasporto della luce sono comuni a molte BRDF. Uno di questi è la *riflessione speculare*. In questo caso la distribuzione della radianza incidente è la stessa della radianza entrante, ma attenuata e riflessa secondo la legge fisica della riflessione. Alcune superfici hanno solamente questo comportamento senza essere diffusive, ad esempio gli specchi. Le superfici diffusive invece riflettono la luce entrante da una singola direzione verso un insieme di direzioni: la loro distribuzione è differente a seconda delle caratteristiche delle BRDF. Nel caso che questa sia costante, allora la distribuzione apparirà uniforme (si dice che è *idealmente diffusiva* o *Lambertiana*). Diversamente si avranno delle variazioni nella distribuzione con dei *picchi* che

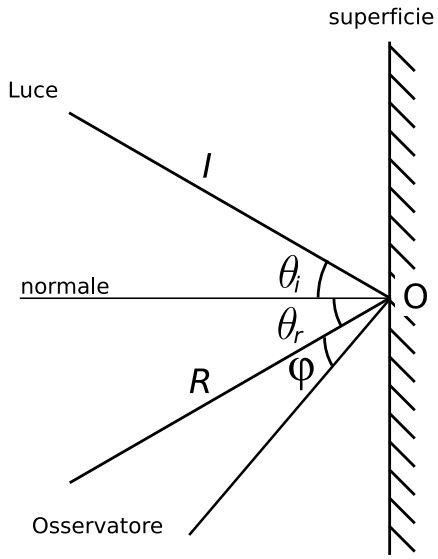


Figura 2.2: *Diagramma della riflessione speculare.*

corrispondono alla direzione speculare e determinano i cosidetti *highlights*. Come si può vedere dalla figura 2.2, se il raggio di luce I che incide su un punto della superficie forma un angolo θ_i con la normale al punto, allora per la legge della riflessione questo sarà uguale all'angolo formato dal raggio riflesso R con la normale al punto, cioè $\theta_i = \theta_r$. Più l'angolo ϕ (compreso tra il raggio riflesso e l'osservatore) è piccolo e tende a zero più sarà visibile un *highlight*.

In figura 2.3 si può vedere in alto la rappresentazione grafica di una BRDF *lambertiana*: la luce colpisce il punto $\mathbf{x}$ della superficie da una direzione ω_i ed è riflesso nella direzione ω_o in maniera uguale per tutte le direzioni. Nella figura sottostante si può vedere una BRDF non lambertiana, che riflette la luce diversamente a seconda della direzione uscente.

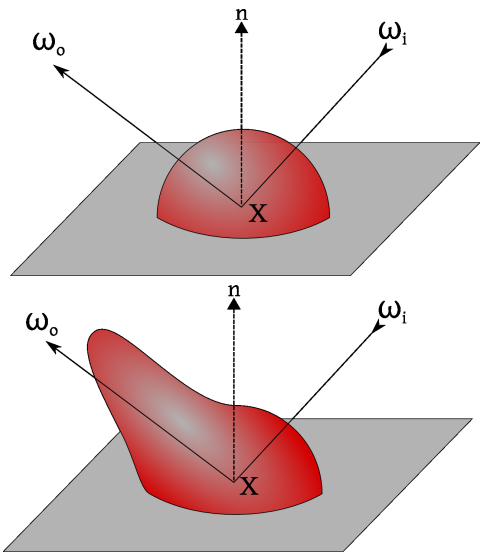


Figura 2.3: *Una brdf idealmente diffusiva (sopra) e non idealmente diffusa (sotto).*

2.2 Stima dell'ambiente di illuminazione attraverso un insieme di basi di funzioni

Marschner[13], il cui lavoro è stato la principale fonte di ispirazione per questa tesi, descrive un metodo per determinare l'illuminazione usata in una fotografia (o più precisamente la luce che può essere fornita come input ad un software di rendering per avere un'output simile alla fotografia) avendo come input la fotografia stessa, la geometria della scena e la BRDF.

Marschner fa alcune assunzioni per ottenere una soluzione univoca. In generale la luce può provenire da ogni punto dello spazio, anche dalla superficie stessa dell'oggetto fotograto, ad esempio nel caso in cui questo sia incandescente. Tuttavia questo caso rende impossibile distinguere un'oggetto di questo tipo da uno che riflette la luce proveniente da una sorgente esterna; dato che casi del genere sono piuttosto rari, l'esclusione di oggetti che emettono luci non inficia l'applicabilità pratica dell'algoritmo. Questa prima

approssimazione non basta per poter stimare il campo radiante incidente su ogni punto della superficie, descritto da una funzione di 4 parametri (l'azimuth e l'elevazione della luce entrante e della direzione di vista). Si è quindi assunto che tutte le luci provenissero da un punto molto lontano rispetto alla dimensione degli oggetti che si stanno guardando, ossia che la radianza incidente è la stessa su ogni punto della superificie.

Sotto queste ipotesi, si può notare che il processo di rendering, fissati i materiali e la geometria che descrivono gli oggetti della scena, definice una funzione lineare che ha come dominio l'illuminazione di sfondo L_b, ossia la luce, e come codominio la radianza riflessa dalla superficie, L_e. La linearità si può anche spiegare in maniera più intuitiva con il principio di *sovrapposizione della luce*: se illuminiamo un oggetto con più sorgenti allo stesso tempo, l'immagine risultante è uguale alla somma delle immagini che risulterebbero se l'oggetto venisse illuminato da ogni sorgente separatamente. In questo modo è possibile incapsulare il processo di rendering sotto una particolare condizione di illuminazione come un singolo operatore lineare $\mathbf{R}$, il quale contiene anche i termini matematici che descrivono la geometria ($\mathbf{G}$) e le proprietà dei materiali ($\mathbf{K}$) e mappa una configurazione delle luci (L_b) in una distribuzione della radianza diffusa(L_e).

$$\mathbf{R} : L_b \rightarrow L_e$$

Uno dei parametri noti, la fotografia, contiene per ogni pixel la misura della radianza riflessa da una piccola area di superficie entro un piccolo cono di direzioni. L'illuminazione viene rappresentata come la combinazione lineare di *un'insieme di basi* ${L_b}^1, \ldots, {L_b}^n$ moltiplicate da apportuni coefficienti (da stimare), ottenendo $L_b = \sum_{i=1}^{n} x_i {L_b}^i$.

Marschner definisce le basi di illuminazione come una partizione di una sfera divisa in regioni triangolari; ogni base è uguale a uno nella corrispon-

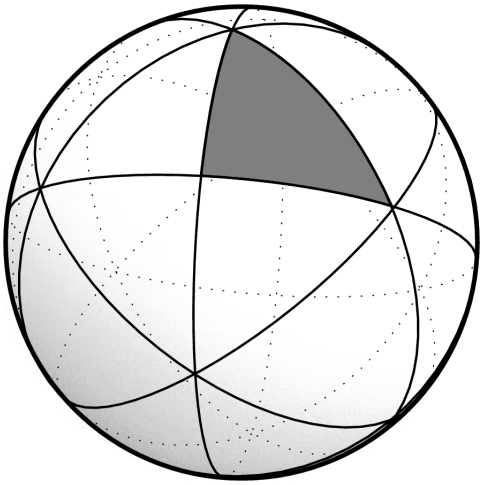


Figura 2.4: *Partizione dell'insieme delle basi, tratta da Marschner [13].*

dente regione (chiamata *regione di supporto*) e zero al di fuori di questa (vedi figura 2.4).

Introducendo un termine K_j, che fornisce il valore del pixel j in funzione di L_e, si può riassumere quanto detto fin'ora in termini matematici:

$$b_j = K_j \mathbf{R} L_b \tag{2.8}$$

che può essere espanso in

$$b_j = \sum_{i=1}^{n} x_i K_j \mathbf{R} {L_b}^i \tag{2.9}$$

Siccome $K_j \mathbf{R} {L_b}^i$ è uno scalare, si può vedere la precedente formula del processo di rendering come un sistema lineare del tipo $Ax = b$, dove A è una matrice $m \times n$, con m uguale al numero dei pixel e n pari al numero delle funzioni di base per l'illuminazione. Avendo come dati noti sia b_j, ossia i pixel della fotografia reale, che A, si possono stimare i coefficienti moltiplicativi per le basi delle luci.

In generale $m \gg n$, poichè ci sono più pixel nella foto che funzioni di base per le luci, quindi si ottiene un sistema sovradeterminato. Inoltre, quando la

BRDF della superficie è regolare , ad esempio se è di tipo Lambertiano, la matrice A è malcondizionata, essendo praticamente singolare. Questo malcondizionamento rende molto sensibile la soluzione al problema del rumore che si verifica nell'acquisizione della fotografia reale. Marschner cerca di risolvere il problema utilizzando una tecnica di *regolarizzazione*, introducendo un termine $\beta(x)$ che misura quanto la soluzione si discosta da una soglia plausibile, minimizzando $\| Ax - b \|^2 + \lambda^2\beta(x)$ invece di $\| Ax - b \|^2$. λ è un parametro, a scelta dell'utente, che media tra quanto Ax approssima b e quanto x è "ragionevole". Il sistema viene quindi risolto tramite decomposizione a valori singolari (SVD, Singular Value Decomposition), più stabile nella risoluzione di sistemi lineari con malcodizionamento rispetto ad altri metodi numerici quali ad esempio Gauss-Jordan.

2.3 Stima dell'illuminazione tramite le ombre proiettate da un oggetto

Sato e Ikeuchi[21] invece presentano una metodo per stimare l'ambiente di illuminazione di una scena a partire dalla luminosità e dalle ombre proiettate da un oggetto sulla stessa scena. Per compensare l'insufficienza di informazioni necessarie alla stima, gli autori usano le informazioni di occlusione della luce causate da un oggetto nella scena così come la luminosità sulla superficie di un oggetto. Più specificamente l'illuminazione viene stimata dai valori di luminosità nelle ombre proiettate da un oggetto di *forma nota* nella scena. Inizialmente la distribuzione dell'illuminazione viene assunta in maniera approssimata come un campionamento discreto di una sorgente di luce estesa (sotto forma di un insieme di sorgenti puntiformi ugualmente distribuite nella scena). Similmente all'approccio di Marschner, viene costruito un sistema lineare dove ogni equazione viene definita da un pixel all'interno di una zona

d'ombra e che ha per incognite i valori della radianza delle sorgenti di luce. Per ottenere questo, Sato e Ikeuchi considerano una *surface image*, ossia un'immagine che comprende un *occluding object*, il quale proietta la proprio ombra su una superficie, chiamata la *shadow surface*.

Anche in questo caso sono state assunte delle ipotesi per poter trovare una soluzione univoca anche se approssimata:

- La geometria dell'*occluding object* e della *shadow surface* è nota.
- L'illuminazione è significativamente distante, ossia i raggi proiettati sugli oggetti della scena sono considerati paralleli
- Non c'è nessuna interiflessione tra gli oggetti presi in considerazione, ossia la luce non si riflette tra l'occlusion object e la shadow surface. Per ottenere questo gli autori considerano come occlusion object degli oggetti opachi scuri con bassa riflessione e alto assorbimento.

A differenza di Marschner, Sato e Ikeuchi non assumono nota la BRDF della shadow surface, ma presentano un metodo per stimare in maniera iterativa sia la distribuzione dell'illuminazione sia la BRDF. Al fine di prendere in considerazione l'illuminazione da tutte le direzioni, viene considerata una patch infinitesima di grandezza $\delta\theta_i$ per $\delta\phi_i$ (espresse in coordinate polari) di una sorgente di luce estesa . Questa, vista da un punto centrale A, sottende un angolo solido $\delta\omega = \sin\theta_i\delta\theta_i\delta\phi_i$ (come è possibile vedere in figura 2.5).

Considerando $L_0(\theta_i, \phi_i)$ come la radianza per unità di angolo solido che proviene dalla direzione (θ_i, ϕ_i), l'irradianza totale nel punto A della superficie è

$$E = \int_{-\pi}^{+\pi} \int_{0}^{\frac{\pi}{2}} L_0(\theta_i, \phi_i) \cos\theta_i \sin\theta_i \delta\theta_i \delta\phi_i.$$

mentre l'occlusione della luce incidente data dall'occluding object è considerata come

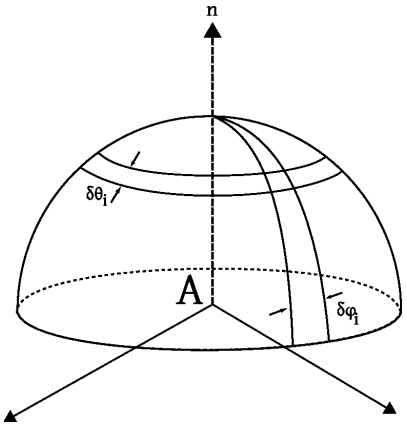


Figura 2.5: *Porzione di angolo solido.*

$$E = \int_{-\pi}^{+\pi} \int_{0}^{\frac{\pi}{2}} L_0(\theta_i, \phi_i) S(\theta_i, \phi_i) \cos\theta_i \sin\theta_i \delta\theta_i \delta\phi_i.$$

dove $S(\theta_i, \phi_i)$ è un termine che tiene proprio conto dell'occlusione, e vale 0 se $L_0(\theta_i, \phi_i)$ è occluso dall'occluding object, altrimenti $S(\theta_i, \phi_i) = 1$.

Quindi la radianza uscente $R_s(\theta_e, \phi_e)$ vista dalla direzione $L_0(\theta_e, \phi_e)$ vale

$$R_s(\theta_e, \phi_e) = \int_{-\pi}^{+\pi} \int_{0}^{\frac{\pi}{2}} f_r(\theta_i, \phi_i, \theta_e, \phi_e) L_0(\theta_i, \phi_i) S(\theta_i, \phi_i) \cos\theta_i \sin\theta_i \delta\theta_i \delta\phi_i. \tag{2.10}$$

dove $f_r(\theta_i, \phi_i, \theta_e, \phi_e)$ è la BRDF. La radianza illuminante della scena è legata con l'irradianza dell'immagine; poichè quello che in realtà osseriviamo non è l'irradianza dell'immagine, ma un valore del pixel registrato nella shadow image, è necessario considerare la conversione tra l'irradianza nel valore di un pixel che corrisponde ad un punto dell'immagine, il quale è ottenuto tramite il sistema di lenti che focalizza un insieme di raggi in una particolare area del CCD. Tuttavia nel fare questo confronto è necessario considerare la distorsione introdotta dal CCD della fotocamera che ha acquisito l'immagine e le conversioni analogico/digitali. Per fare questo, basandosi su altri studi, gli autori notano che i due valori da confrontare, la radianza uscente $R_s(\theta_e, \phi_e)$

e l'irradianza della fotografia $P(\theta_e, \phi_e)$ sono in realtà proporzionali ad un fattore k ottenuto dalla calibrazione del CCD della macchina fotografica. Di conseguenza la (2.10) diventa

$$P(\theta_e, \phi_e) = k \int_{-\pi}^{+\pi} \int_{0}^{\frac{\pi}{2}} f_r(\theta_i, \phi_i, \theta_e, \phi_e) L_0(\theta_i, \phi_i) S(\theta_i, \phi_i) \cos\theta_i \sin\theta_i \delta\theta_i \delta\phi_i. \tag{2.11}$$

dove $P(\theta_e, \phi_e)$ è il valore del pixel nella shadow image, e k è il fattore di scala tra la radianza della scena e il valore del pixel.

In questa maniera per poter risolvere il sistema lineare avendo come incognita la radianza $L_0(\theta_i, \phi_i)$, la distribuzione dell'illuminazione viene approsimata attraverso un campionamento discreto dell'intera superficie della sorgente di luce estesa, ad esempio approssimando l'equazione 2.11 con

$$P(\theta_e, \phi_e) = k \sum_{i=1}^{n} f_r(\theta_i, \phi_i, \theta_e, \phi_e) L_0(\theta_i, \phi_i) S(\theta_i, \phi_i) \cos\theta_i \sin\theta_i \tag{2.12}$$

con n pari al numero delle direzioni del campionamento. In seguito vengono definiti approcci differenti a seconda di quale BRDF è data. Nel caso di una BRDF lambertiana, poichè questa è costante, l'equazione 2.12 si riduce a

$$P(\theta_e, \phi_e) = k \sum_{i=1}^{n} K_d L_0(\theta_i, \phi_i) S(\theta_i, \phi_i) \cos\theta_i \sin\theta_i \tag{2.13}$$

dove K_d è il parametro di riflessione diffusa della superficie lambertiana. Da 2.13 si ricava il valore P del pixel dell'immagine

$$P = \sum_{i=1}^{n} a_i L_i \tag{2.14}$$

dove L_i sono n radianze di luci sconosciute, mentre gli a_i rappresentano il restante termine della 2.13 e sono noti. Selezionando un numero opportuno k di pixel si ottiene un sistema di equazioni lineari

$$P_j = \sum_{i=1}^{n} a_{ji} L_i \qquad j = 1, \ldots, k. \tag{2.15}$$

Per selezionare i pixel più significativi e non ridondanti, al fine di avere un sistema lineare stabile numericamente, gli autori partizionano la *shadow surface* in *cluster*, dove ogni cluster è composto dai pixel che hanno la stessa combinazione di a_i. k è scelto opportunamente per risolvere il sistema per un insieme di L_i sconosciuti. E' importante notare come il fattore $S(\theta_i, \phi_i)$ sia fondamentale nella scelta dei coefficienti, in quanto la variazione di a_{ji} aumenta quando $S(\theta_i, \phi_i)$ cambia significativamente.

2.4 Illuminazione globale inversa per la stima di BRDF a partire da fotografie

Devebec e al. [27], presentano invece un metodo per stimare la BRDF degli oggetti di una scena reale a partire da un insieme di fotografie, prendendo in considerazione sia l'illuminazione diretta che quella indiretta e partendo sempre dai valori della radianza delle fotografie. L'innovazione nel loro metodo consiste nell'uso di un insieme limitato di fotografie, riprese solo da un limitato numero di angoli, piuttosto che un insieme denso di fotografie riprese per ogni punto di ogni superficie da un vasto range di angoli. In particolare per poter avere delle stime accurate è necessario che le fotografie coprano per ogni oggetto almeno un'area con una riflessione speculare alta (un *highlight*) e un'area con una bassa componente speculare. Questi dati tuttavia sono troppo esigui per poter stimare la BRDF: la sua stima non è la medesima che si otterrebbe con un gonioreflettometro, ma è basata su pochi parametri. Inoltre si assume che la scena possa essere partizionata in aree con le stesse caratteristiche di riflessione, consentendo che la riflessione diffusa, o *albedo*, dell'oggetto vari arbitrariamente su ogni superficie. L'albedo stimata viene calcolata in una mappa chiamata *albedo map*. Invece le proprietà di riflessione direzionali (come la specularità) vengono considerate costanti su ogni

area.

In una scena reale, le superfici normalmente si illuminano a vicenda, ossia la luce che una superficie riceve non arriva soltanto dalle sorgenti di luce, ma anche dagli altri oggetti attraverso l'illuminazione indiretta. La radianza incidente è una funzione piuttosto complessa che ha come parametri le sorgenti di luce, la geometria della scena, e la BRDF della superifici da stimare. In questo lavoro gli autori usano i dati della radianza dalle fotografie e tecniche di *image based rendering* per stimare la radianza incidente sulle superifici nella scena.

2.4.1 Inverse Radiosity

Poichè molte superfici hanno riflessioni sia di tipo diffusivo che di tipo speculare (come spiegato in 2.1.2), la stima di entrambe è difficile in un ambiente dove sono presenti interriflessioni tra oggetti oppure oggetti trasparenti, quindi vengono considerate soltanto superfici opache di tipo diffusivo Lambertiano. Gli autori definiscono l'*inverse radiosity* per stimare il *diffuse albedo* (la riflessione diffusa, indipendente dalle condizioni di illuminazione) avendo a disposizione la geometria, l'illuminazione e la distribuzione della radianza nella scena (l'insieme di fotografie). Nel radiosity "diretto" (come accennato nell'introduzione) le superfici sono suddivise in un numero appropriato di *patches*, tale da considerare il termine di radiosità (cioè il flusso di luce che esce da una superificie) e l'albedo come costanti.

Per ogni patch vale

$$B_i = E_i + \rho_i \sum_j B_j F_{ij} \tag{2.16}$$

dove B_i, E_i e ρ_i sono rispettivamente la radiosità, l'emissione e l'albedo diffusa della patch i , mentre F_{ij} è un *form factor* (che dipende dalla geometria della scena) tra le patch i e j, ossia una proporzione dell'energia totale che

viene riflessa da i verso j. Poichè le superfici Lambertiane hanno una distribuzione uniforme della radianza, questo rende noti i parametri B_i, E_i, mentre i F_{ij} possono essere derivati dalla geometria. In questo modo è possibile ricavare $\rho_i = (B_i - E_i)/(\sum_j B_j F_{ij})$

2.4.2 Stima di una BRDF parametrica dall'illuminazione diretta

Per stimare la BRDF in una forma parametrica (non estesa e accurata come quella restituita da un gonioreflettometro), viene considerata una superficie con BRDF uniforme illuminata da una sorgente di luce puntiforme e fotografata da una fotocamera. Ogni pixel di una fotografia fornisce una misura della radianza L_i del corrispondente punto P_i della superficie in direzione della camera. La posizione della luce permette di calcolare l'irradianza I_i incidente sul punto P_i. L'obiettivo degli autori è di usare i dati noti (L_i,I_i) per stimare la BRDF della superificie. Essendo questa una funzione di 4 variabili (l'azimuth e l'elevazione della luce entrante e della direzione di vista), sono necessari più di un insieme bidimensionale per avere una stima accurata. Tuttavia per molti materiali è possibile approssimare la BRDF attraverso un modello con un piccolo numero di parametri. In questo caso viene usato il modello di Ward ([25]) nel quale la BRDF è modellata come la somma di un termine diffusivo ρ_d e uno speculare ρ_s. Questo porta alla definizione di un'equazione per ogni punto P_i della superficie, del tipo:

$$L_i = (\rho_d + \rho_s) I_i \tag{2.17}$$

Nel sistema ci sono tante equazioni quanti sono i pixel nella parte della fotografia corrispondente alla patch della superficie, quindi è necessario risolvere un problema di ottimizzazione non lineare per trovare la stima migliore di ρ_d e ρ_s.

2.4.3 Stima di una BRDF parametrica dall'illuminazione diretta e indiretta

Nel caso in cui si prenda in considerazione sia l'illuminazione diretta che quella indiretta, l'equazione 2.17 si complica. Prendendo in considerazione un punto P_i su una patch A_i inquadrata da una camera C_v, la radianza da P_i in direzione della camera è la riflessione della luce incidente, costituita dalla somma di tutte le sorgenti di luce (la parte diretta) più quella delle superfici intorno (la parte indiretta). L'equazione 2.17 assume quindi una forma più generale

$$L_{C_v P_i} = E_{C_v P_i} + \rho_d \sum_j L_{P_i A_j} F_{P_i A_j} + \rho_s \sum_j L_{P_i A_j} K_{C_v P_i A_j} \tag{2.18}$$

dove $L_{C_v P_i}$ è il valore della radianza verso la camera C_v del punto P_i, $E_{C_v P_i}$ è l'emissione del punto verso la camera, $L_{P_i A_j}$ è il valore della radianza dalla patch A_j verso P_i mediato dal *form factor* $F_{P_i A_j}$ e moltiplicato per il coefficiente diffusivo ρ_d, mentre $K_{C_v P_i A_j}$ è il termine speculare rispetto alla camera C_v e una "sorgente di luce" (indiretta) posta in A_j. Come prima, l'obiettivo è la stima di ρ_d e ρ_s. Nell'equazione $E_{C_v P_i} = 0$ per tutte le superfici che non emettono luce, $L_{C_v P_i}$ è noto dai pixel della fotografia (ripresa dalla fotocamera C_v), però $L_{P_i A_j}$ non può essere misurato direttamente, ma deve essere stimato iterativamente, supponendo ad esempio che la patch A_j appaia in un'altra fotografia ripresa da una camera C_k e assumendo che A_j sia di tipo lambertiano (in questo caso caso $L_{P_i A_j}$ diventerebbe equivalente a $L_{C_k A_j}$, poichè le superfici lambertiane hanno una radianza costante non dipendente dal punto di vista). Uno schema dell'interazione descritta è proposto in figura 2.6.

Per poter fare queste assunzioni, è necessario avere un'insieme di foto che partizionino gli oggetti della scena e ogni patch necessita di un camera dalla quale viene selezionata la fotografia. Inoltre per poter stimare la spe-

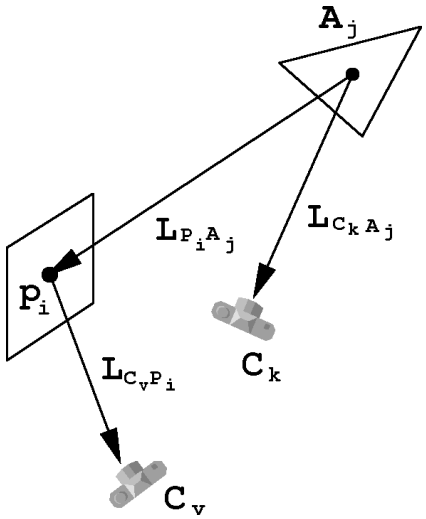


Figura 2.6: *Interazione della luce tra due diverse patch, immagine tratta da Debevec et al. [27]*

cularità, almeno un riflesso speculare deve essere visibile nell'insieme delle immagini. Come nel caso della sola luce diretta viene risolto un problema di ottimizzazione non lineare.

2.4.4 Stima dell'albedo map

L'albedo su ogni punto x della superficie è definito come

$$\rho_d(x) = \pi \frac{D(x)}{I(x)} \tag{2.19}$$

dove $\rho_d(x)$ è la *diffuse albedo map*, $D(x)$ è la mappa della radianza diffusiva e $I(x)$ è la mappa dell'irradianza. Supponendo di avere un'immagine che copre una superficie la cui mappa della radianza è $L(x) = D(x) + S(x)$, dove $S(x)$ è la mappa della radianza speculare vista dalla posizione della camera. Molto semplicemente, da quest'ultima osservazione si può ricavare $D(x) = L(x) - S(x)$ e, usando i parametri della riflessione speculare che sono stati calcolati nella fase precedente, ottenere l'albedo map.

2.5 Stima analitica della radianza tramite deconvoluzione

Ramamoorthi et al. [20] pone le basi per un *framework* matematico per il rendering inverso. Nei lavori citati precedentemente, spesso i metodi erano limitati da condizioni particolari di illuminazione; in questo lavoro gli autori introducono un framework che descrive il campo di luce riflessa come una *convoluzione* tra l'illuminazione e la BRDF, attraverso l'uso di *spherical harmonics*, basandosi sul lavoro di Bastos et al [3]. Il rendering inverso viene quindi visto come una *deconvoluzione*. L'obiettivo degli autori è applicare il rendering inverso ad un ambiente di illuminazione non controllato e di poter stimare più di un termine sconosciuto *simultaneamente* (ad esempio BRDF e illuminazione). Anche in questo caso le assunzioni riguardano la geometria, l'illuminazione, per la quale sono considerate sorgenti di luce distanti (consentendo di trattare l'illuminazione come una funzione dell'angolo incidente); le BRDF, che sono isotropiche, ossia non cambiano rispetto ad una rotazione della superficie intorno alla sua normale.

2.5.1 Interpretazione come convoluzione

Nel dominio spaziale, la convoluzione è generata quando un filtro è traslato su un segnale di input, ossia

$$(f \otimes g)(t) = \int f(\tau)g(t-\tau)\,d\tau \tag{2.20}$$

La BRDF può essere vista come un "filtro", mentre l'ambiente di illuminazione è il segnale di input. Il campo di luce riflesso può quindi essere visto tramite la convoluzione della BRDF con l'illuminazione, ossia filtrando quest'ultima tramtite la BRDF. La convoluzione avviene in questo caso nel dominio angolare e le basi per derivarla in tale dominio sono dette *spherical*

harmonics, le quali sono le analoghe sulla sfera delle basi di Fourier sulla linea o il cerchio. Esse sono trattate in modo approfondito da Inui et al. [11].

Tramite questo framework, gli autori concludono che la stima di una BRDF è ben condizionata quando l'illuminazione contiene altre frequenze (cioè tante luci direzionali), mentre è malcondizionata nel caso di illuminazione diffusa. La stima dell'ambiente di illuminazione invece è ben condizionato quando la BRDF ha componenti ad alta frequenza, come riflessi speculari accentuati, mentre è malcondizionato quando è di tipo diffusiva (ad esempio una BRDF di tipo Lambertiano). In particolare, un materiale Lambertiano agisce come un filtro passa-basso, impedendone la stima delle componenti ad alta frequenza.

Capitolo 3

Framework per rendering inverso basato su Wavelet

3.1 Tecnologie e metodi utilizzati

Prima di descrivere in profondità il funzionamento dell'algoritmo per la stima dell'ambiente di illuminazione qui proposto è necessario introdurre alcune nozioni teoriche utilizzate, come le *wavelet*, le *high dynamic range images* e il *Montecarlo raytracing*.

3.1.1 Wavelet

Le *wavelet* sono uno strumento matematico per la rappresentazione di un segnale tramite una suddivisione gerarchica di funzioni. Esse consentono di descrivere il segnale a diversi livelli di dettaglio e sono quindi molto adatte per rappresentare segnali in multirisoluzione, ad esempio per comprimere le immagini. L'idea fondamentale alla base delle wavelet riguarda l'analisi dei segnali tramite scalatura e traslazione. In realtà quest'idea non è nuova, poichè l'approssimazione di funzioni tramite funzioni *base* è stata definita per prima da Joseph Fourier, ma nel caso delle wavelet il fattore di scala gioca un ruolo chiave. Infatti, poichè gli algoritmi che eseguono un analisi di tipo wavelet processano i dati a differenti risoluzioni, se si effettua un analisi

Figura 3.1: *A sinistra l'immagine originale, a destra la stessa immagine compressa tramite wavelet.*

con una finestra "larga" (cioè con dettaglio basso), si noteranno soltanto le caratteristiche più evidenti del segnale; usando una finestra "piccola" (cioè un dettaglio più alto) si noteranno le discontinuità più fini. Per esempio, nel caso della compressione di immagini in figura 3.1 a sinistra è mostrata l'immagine originale mentre a destra un'immagine compressa tramite wavelet (27:1) con un basso livello di dettaglio.

In questa tesi lo strumento delle wavelet è usato come base di funzioni per l'ambiente di illuminazione; infatti questo può essere rappresentato attraverso una parte a risoluzione più bassa data dall'illuminazione diffusa, ad esempio un luce che ricopre un'ampia superficie e illumina estensivamente l'oggetto e una un parte a risoluzione più alta, ad esempio una piccola luce puntiforme posta in un punto specifico dell'ambiente. Prima di trattare le wavelet nel caso bidimensionale, è utile trattare inizialmente il caso ad una dimensione.

Wavelet in una dimensione

Si può vedere un segnale monodimensionale come una sequenza di coefficienti o alternativamente come una funzione nell'intervallo $[0, 1)$. Un segnale è una funzione *costante a tratti* su $[0, 1)$.

Sia V^0 lo spazio vettoriale di tutte queste funzioni. Similmente, una funzione costante sugli intervalli $[0, 1/2)$ e $[1/2, 1)$ apparterrà a V^1. In generale

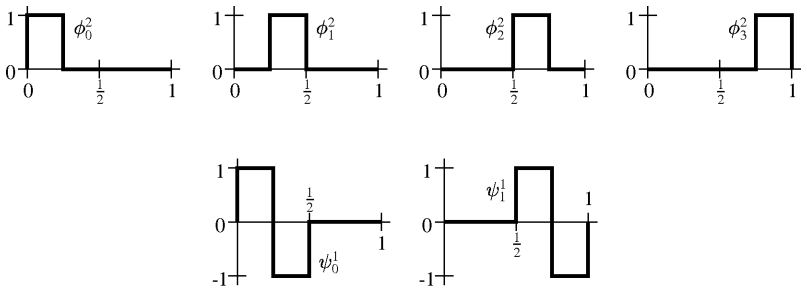


Figura 3.2: *La funzione ϕ_i^j e ψ_i^j.*

una funzione costante a tratti con 2^j intervalli sarà contenuta in V^{j-1}. Siccome tutte le funzioni sono contenute nell'intervallo unitario, ogni vettore di V^j è anche contenuto in V^{j+1}, ottenendo una successione V^i crescente

$$V^0 \subset V^1 \subset V^2 \subset \dots \tag{3.1}$$

E' quindi necessario definire una base per ogni spazio vettoriale V^j. Le basi di funzioni di ogni spazio V^j sono denotate con il simbolo ϕ e sono chiamate *funzioni di scala*. Se consideriamo una funzione molto semplice, chiamata funzione "box", definita come

$$\phi(x) = \begin{cases} 1 & \text{se } 0 \leqslant x < 1 \\ 0 & \text{altrimenti} \end{cases} \tag{3.2}$$

e le applichiamo una traslazione e scalatura

$$\phi_i^j(x) = \phi(2^j x - i) \qquad i = 0, \dots, 2^j - 1, \tag{3.3}$$

abbiamo una base per lo spazio V^j. Una sua rappresentazione grafica è mostrata in figura 3.2.

Si definisce quindi un nuovo spazio vettoriale W^j come *complemento ortogonale di V^j in V^{j+1}*, cioè W^j è lo spazio delle funzioni in V^{j+1} che sono ortogonali a tutte le funzioni in V^j. Si chiamano *wavelet* le funzioni $\psi_i^j(x)$

linearmente indipendenti che formano uno *span*[1] di W^j. Queste funzioni hanno due importanti proprietà:

1. Le basi $\psi_i^j(x)$ di W^j, insieme con $\phi_i^j(x)$ di V^j formano una base di V^{j+1}.
2. Ogni base $\psi_i^j(x)$ di W^j è ortogonale ad ogni funzione di $\phi_i^j(x)$ di V^j.

Le wavelet che corrispondono alla funzione "box" definita sopra sono le wavelet di *Haar* e sono definite a partire dalla funzione ψ, la quale viene anche chiamata "mother wavelet":

$$\psi(x) = \begin{cases} 1 & \text{se } 0 \leqslant x < 1/2 \\ -1 & \text{se } 1/2 \leqslant x < 1 \\ 0 & \text{altrimenti} \end{cases} \tag{3.4}$$

come

$$\psi_i^j(x) = \psi(2^j x - i) \qquad i = 0, \ldots, 2^j - 1, \tag{3.5}$$

Inoltre le basi di Haar possiedono la proprietà di essere ortogonali tra loro.

Wavelet in due dimensioni

Nel caso bidimensionale, ad esempio considerando come segnale un'immagine della quale si devono trasformare i pixel, esistono due modi per poter effettuare una decomposizione tramite wavelet. Nel primo caso si applica la trasformata appena vista ad ogni riga della matrice di pixel, ottenendo un valore medio (dato dalla ϕ) con i coefficienti di dettaglio per ogni riga (dati dalle ψ); quindi si applica la medesima trasformazione alle colonne. Nell'altro caso, chiamato *decomposizione nonstandard*, si alterna tra trasformazioni sulle righe e sulle colonne. La costruzione non standard, che è il metodo adottato

[1] siano $v_1, ..., v_r \in V$, allora lo span $(v_1, ..., v_r) = \{\lambda_1 v_1 + \cdots + \lambda_r v_r | \lambda_1, \ldots, \lambda_r \in \mathbb{K}\}$

in questa tesi, ed è anche quello comunemente adottato per la compressione di immagini, procede definendo una funzione di scala bidimensionale

$$\phi\phi(x,y) = \phi(x)\phi(y) \tag{3.6}$$

e tre funzioni wavelet

$$\phi\psi(x,y) = \phi(x)\psi(y) \tag{3.7}$$

$$\psi\phi(x,y) = \psi(x)\phi(y) \tag{3.8}$$

$$\psi\psi(x,y) = \psi(x)\psi(y) \tag{3.9}$$

Ogni wavelet misura le variazioni su differenti direzioni: $\phi\psi$ riporta le variazioni lungo le righe, $\psi\phi$ lungo le colonne e $\psi\psi$ lungo le diagonali. Come nel caso ad una dimensione, si denotano i livelli di scala con l'indice j e le traslazioni orizzontali e verticali con gli indici k e l. La *base non standard* viene rappresentata da una funzione $\phi\phi^0_{0,0} = \phi(x)\phi(y)$ e dalle tre funzioni wavelet traslate e scalate

$$\phi\psi^j_{k,l}(x,y) = 2^j\phi\psi(2^jx-k, 2^jy-l) \tag{3.10}$$

$$\psi\phi^j_{k,l}(x,y) = 2^j\psi\phi(2^jx-k, 2^jy-l) \tag{3.11}$$

$$\psi\psi^j_{k,l}(x,y) = 2^j\psi\psi(2^jx-k, 2^jy-l) \tag{3.12}$$

In figura 3.3 è possibile vedere una rappresentazione della costruzione non standard delle basi di Haar per V^2 nel caso bidimensionale.

Compressione tramite wavelet

Come anticipato in 3.1.1 le wavelet sono indicate per la compressione, ossia per approssimare un insieme di dati usandone un insieme più piccolo. Infatti in generale l'occupazione di dati multimediali come immagini o video in forma non compressa richiede una considerevole quantità di spazio e ampiezza di banda. Per esempio un video di 1 minuto a 30 fotogrammi al secondo alla

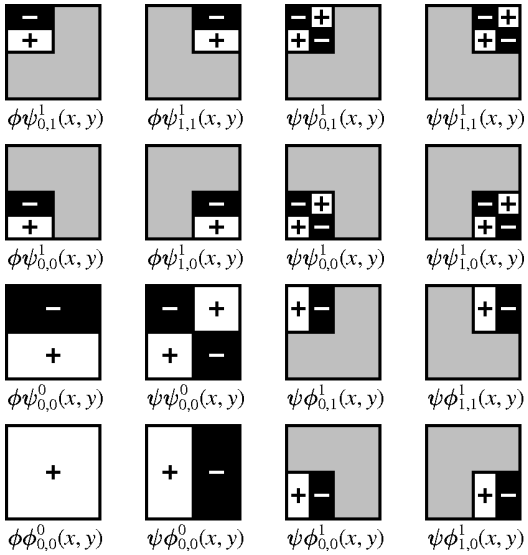


Figura 3.3: *Costruzione non standard delle basi di Haar per* V^2

risoluzione di 640 × 480 a 24 bit per pixel richiederebbe 1.66 Gigabyte di dati. Nel caso delle immagini si può notare che spesso i pixel vicini tra loro sono correlati e quindi contengono informazioni ridondanti. Per quanto riguarda il nostro caso, le wavelet sono usate per comprimere la funzione che stima l'ambiente di illuminazione; la ridondanza è dovuta ad esempio alla vicinanza di due luci, le quali illuminano in maniera equivalente il modello, oppure luci molto piccole che non lo illuminano per niente e quindi possono essere scartate. La compressione wavelet è di tipo *Transform coding*, cioè trasforma il segnale in ingresso dal dominio spaziale ad un altro tipo di rappresentazione e quindi codifica i valori trasformati. L'idea alla base della trasformata wavelet è la rappresentazione di ogni funzione arbitraria $f(x)$ come la somma di funzioni di base wavelet. Infatti, se la funzione $f(x)$ è esprimibile come la somma pesata di funzioni base u_i

$$f(x) = \sum_{i=0}^{n} c_i u_i(x) \tag{3.13}$$

allora l'insieme dei dati è rappresentato dai coefficienti c_i. E' quindi necessario trovare una funzione $g(x)$ che approssimi $f(x)$ richiedendo meno coefficienti e possibilmente considerando anche basi differenti:

$$g(x) = \sum_{i=0}^{m} d_i v_i(x) \tag{3.14}$$

tale che

$$m < n \quad \text{e} \quad \|f(x) - g(x)\| \leq \epsilon \tag{3.15}$$

per un certo valore ϵ di errore di approssimazione.

Se consideriamo le stesse basi, ossia $u_i(x) = v_i(x)$, allora si può vedere il problema della compressione come un problema di ordinamento dei coefficienti, tali che per ogni $m < n$ i primi m elementi della sequenza danno la migliore approssimazione della funzione $f(x)$ secondo una data norma, ad esempio la norma L^2.

Sia σ una permutazione degli indici $1, \ldots, n$ dei coefficienti, ossia una funzione che dato un insieme di indici in ingresso li combina scambiandoli di posizione; più precisamente è una funzione biunivoca $\sigma : D \to D$, dove D è il dominio degli indici; ad esempio, dato l'insieme di indici $\{5, 6, 7, 8\}$, $\sigma^{inv}(\{5, 6, 7, 8\}) = \{8, 7, 6, 5\}$ inverte gli indici.

Sia $g(x)$ una funzione che usa i coefficienti che corrispondono ai primi m numeri della permutazione ($m < n$):

$$g(x) = \sum_{i=1}^{m} c_{\sigma(i)} u_{\sigma(i)}. \tag{3.16}$$

Definendo il prodotto scalare come $\langle f|g\rangle$, l'errore diventa:

$$\begin{aligned}\|f(x)-g(x)\|_2^2 &= \langle f(x)-g(x)|f(x)-g(x)\rangle \\ &= \langle \sum_{i=m+1}^{n} c_{\sigma(i)}u_{\sigma(i)}| \sum_{j=m+1}^{n} c_{\sigma(j)}u_{\sigma(j)}\rangle \\ &= \sum_{i=m+1}^{n}\sum_{j=m+1}^{n} c_{\sigma(i)}c_{\sigma(j)}\langle u_{\sigma(j)}(x)|u_{\sigma(i)}(x)\rangle \\ &= \sum_{i=m+1}^{n} c_{\sigma(i)}^2.\end{aligned} \tag{3.17}$$

essendo le basi u_i ortonormali, come le wavelet di Haar.

Per minimizzare l'errore, risulta quindi che la migliore scelta per la permutazione σ è quella che ordina i coefficienti in ordine di importanza decrescente (ossia vale la relazione $|c_{\sigma(1)}| \geq \ldots |c_{\sigma(n)}|$), come in effetti fanno le wavelet poichè per indici bassi rappresentano i dettagli a grana più grossa mentre per indici più alti rappresentano i vari livelli di dettaglio. Infatti la wavelet di livello 0 rappresenta il valor medio della funzione, mentre i livelli più alti rappresentano i dettagli in ordine di risoluzione sempre più alta.

Uso delle wavelet

Come accennato, le wavelet sono particolarmente indicate per la compressione dei segnali. Come specificato in Ramamoorthi et al. [14] e [23] è necessaria soltanto una percentuale esigua di tutte le basi in cui viene decomposto un segnale; Ramamoorthi usa l'1% delle basi per approssimare un ambiente di illuminazione. Infatti, consideriamo l'equazione che descrive l'illuminazione diretta da parte di un environment map, la quale è un tipo di immagine che descrive un ambiente di illuminazione (descritta in 3.2.1):

$$B(\mathbf{x_i}) = \sum_j T(\mathbf{x_i}, \omega_j)L(\omega_j) \tag{3.18}$$

B è il valore della radianza calcolata, L è il valore dell'illuminazione e T è un fattore di trasporto, cioè

$$T(\mathbf{x}, \omega_j) = S(\mathbf{x}, \omega_j) f_r(\mathbf{x}, \omega \rightarrow \omega_o(\mathbf{x})(\omega \cdot n(\mathbf{x})). \tag{3.19}$$

che codifica il come la luce viene riflessa nella scena, poichè S è la funzione di visibilità che ritorna 1 se una raggio dal punto $\mathbf{x}$ verso la direzione ω è visibile e 0 altrimenti, f_r è la BRDF e $(\omega \cdot n(\mathbf{x}))$ è il coseno dell'angolo incidente.

La 3.18 si può scrivere in notazione matriciale come

$$B = TL, \tag{3.20}$$

In particolare l'equazione vale anche per ogni rappresenatazione di L che sia ortonormale, come nel caso delle wavelet di Haar.

L'autore quindi effettua un confronto tra spherical harmonics (poichè anch'esse sono una base ortonormale) e wavelet; le prime non hanno una buona località spaziale, ossia è necessaria un'insieme ampio di spherical harmonics per rappresentare una piccola *area light*[2] e rappresentano male le discontinuità che queste creano. Una base wavelet invece contiene le area light che variano in grandezza da un pixel della cube environment map fino alla grandezza di tutto il quadrato di un lato dell'environment cube. In figura 3.4 si può vedere come con 200 wavelet, correspondenti a circa l'1% del totale, si riesca a stimare correttamente l'ambiente di illuminazione.

Nel lavoro di questa tesi allo stesso modo vengono scartate le wavelet che non danno un contributo significativo all'illuminazione della scena, tramite delle stime che verranno spiegate in seguito. Poichè il rendering di ogni base wavelet è la parte che praticamente è la più lenta dell'algoritmo (sebbene sia accelerato via hardware grafico), questo permette di ottimizzare la stima riducendo significativamente i tempi di calcolo.

[2]Un'area light è una luce non puntiforme, che occupa un'area estesa e finita

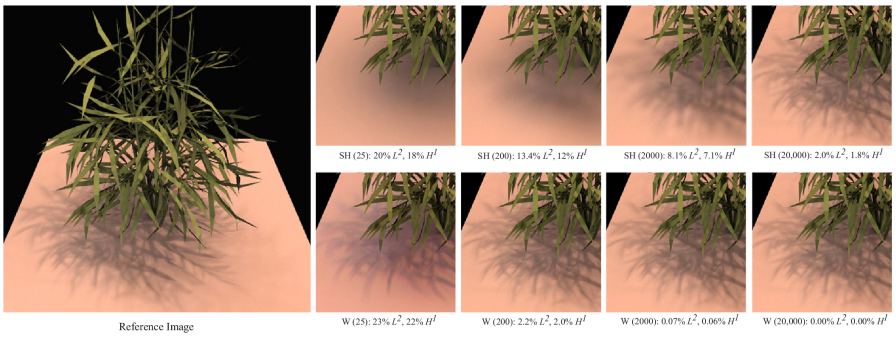


Figura 3.4: *Rappresentazione di un rendering tramite ambiente di illuminazione approssimato con* spherical harmonics*(SM) e* wavelet*(W)*

3.1.2 High Dynamic Ranges Images (HDRI)

Per memorizzare le wavelet bidimensionali in immagini si è fatto uso delle librerie e del formato grafico OpenEXR, sviluppato congiuntamente da Nvidia e Industrial Light and Magic. Questo formato è molto flessibile e permette di memorizzare immagini con ampia gamma dinamica (le cosiddette *High Dynamic Ranges Images*, HDRI) cioè immagini dove i valori RGB non sono memorizzati come interi ad 8 bit, limitati quindi a valori tra 0 e 255, ma come valori *float* permettendo di esprimere un più ampio range di valori ed esposizioni, cioè le differenze tra le aree illuminate e quelle scure, molto ampie (vedi figura 3.5). Infatti l'informazione memorizzata in un'immagine HDRI tipicamente corrisponde al valore della radianza, diversamente dalla immagini digitali tradizionali (dette anche LDRI, cioè Low Dynamic Ranges Images) che rappresentano i colori che dovrebbero apparire su un particolare dispositivo, come un monitor.

Inoltre questo permette di gestire anche numeri negativi, come nel caso dei valori delle wavelet citate sopra. L'output del software di rendering è stato memorizzato in questo formato per evitare di avere una perdità di qualità dovuta alla quantizzazione e per memorizzare proprio il valore corretto della

radianza.

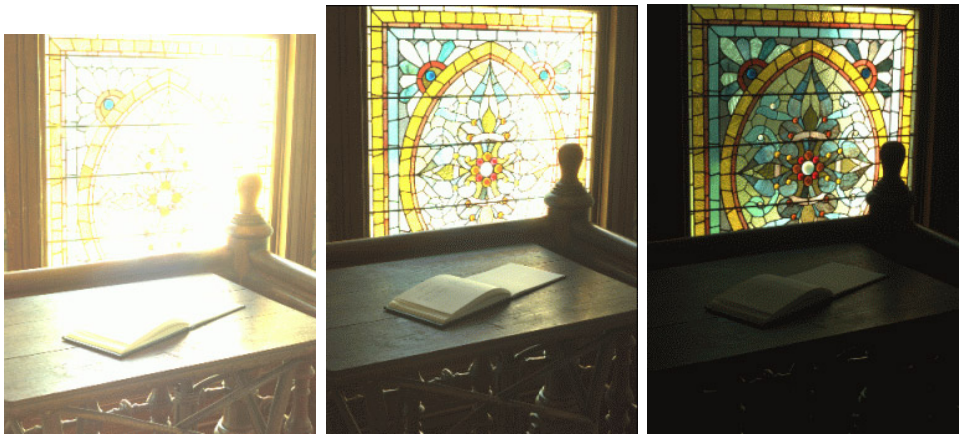


Figura 3.5: *Una HDRI consente di memorizzare all'interno dello stessa immagine più fattori di esposizione fotografica.*

3.1.3 Montecarlo Raytracing

Per risolvere l'equazione di rendering 2.1 sono stati proposti vari algoritmi di tipo *Montecarlo*, usati spesso per simulare processi fisici e sistemi matematici in maniera *stocastica*, ossia tramite comportamenti non deterministici nei quali uno stato non è completamente determinato dal precedente. L'uso di questo metodo deriva da una connessione tra l'integrazione e il *valore atteso* di una variabile aleatoria, infatti un integrale può essere visto come il valore atteso (la media) di una variabile aleatoria:

$$E(f(x)) = \int f(x)p(x)dx \tag{3.21}$$

dove $p(x)$ è la funzione di distribuzione delle probabilità della generica funzione $f(x)$. Molti problemi riguardano la somma di un insieme di variabili *indipendenti* x_i le quali condividono una densità p (le x_i sono chiamate variabili aleatorie indipendenti identicamente distribuite). Quando la somma è

divisa per il numero di variabili, si può ricavare una stima di $E(x)$:

$$E(x) \approx \frac{1}{N} \sum_{i=1}^{N} x_i \tag{3.22}$$

Facendo la media di molti campioni di una variabile aleatoria è quindi possibile approssimare l'integrale. Infatti vale

$$E(f(x)) = \int f(x)p(x)dx \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{3.23}$$

Tuttavia invece che approssimare l'integrale di un prodotto di funzioni $f \cdot p$ è a volte utile effettuare la stessa operazione su un'unica funzione g. Come notato da [2] a questo si può ovviare notando che, se $g = f \cdot p$, allora vale

$$\int g(x)dx \approx \frac{1}{N} \frac{\sum_{i=1}^{N} g(x_i)}{\sum_{i=1}^{N} p(x_i)} \tag{3.24}$$

dove $p(x_i)$ deve essere diverso da zero. Proprio la scelta dei $p(x_i)$ è importante affinchè il rapporto g/p abbia una bassa varianza e quindi l'approssimazione sia più precisa. Il problema è che ci sono infinite variabili aleatorie che possono essere associate ad un dato integrale, ma alcune meglio di altre sono adatte al campionamento. Un'assunzione del metodo Montecarlo è che la funzione di distribuzione della probabilità della variabile aleatoria dovrebbe essere più simile possibile alla funzione integranda.

La forma più diffusa di integrale nel nostro caso è quelle che esprime la radianza riflessa da una superficie, del tipo

$$\int_S f(\omega)(\omega \cdot n)d\omega \tag{3.25}$$

dove S è un sottinsieme della sfera unitaria, ω è un vettore unitario che indica la direzione e n è la normale alla superficie. Per poter applicare il metodo di Montecarlo in maniera efficace è necessario effettuare un campionamento che corrisponda alle distribuzioni delle luci e delle geometrie che si trovano in una

scena. Riprendiamo l'equazione 2.1 per vedere come è possibile applicare il metodo Montecarlo per risolverla

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) d\vec{w}' \tag{3.26}$$

Quest'equazione esprime si applica a tutti i punti di tutte le superfici in una scena; le variabili note sono L_e e la BRDF f_r, poichè dipendono dalla geometria della scena, dalle caratteristiche dei materiali e dalle luci. La variabile non nota è L_o e L_i, essendo entrambe le radianze riflesse dalle superfici. Questa equazione tuttavia si può scrivere in maniera più compatta e leggibile usando gli *operatori*, che sono un metodo per mappare una funzione in un'altra funzione:

$$L = L_e + K \circ L \tag{3.27}$$

Spesso è comodo spezzare K in due operatori T e S.

S rappresenta rispettivamente l'operatore di *scattering* che prende la luce incidente e restituisce la luce uscente, mentre T è l'operatore di *trasferimento* che prende la luce uscente da una superficie e la trasferisce su un'altra. Essendo una formula iterativa l'equazione di rendering può essere svolta in

$$\begin{aligned} L^0 &= L_e \\ L^1 &= L_e + K \circ L = L_e + K \circ L_e \\ &\dots \\ L^n &= L_e + K \circ L^{n-1} = \sum_{i=0}^{n} K^i \circ L_e \end{aligned} \tag{3.28}$$

Notando che $K^0 = I$ (dove I è l'operatore identità) si può notare che questa somma è una serie di Neumann del tipo

$$(1 - x)^{-1} = \sum_{k=0}^{\infty} x^k \tag{3.29}$$

Quindi

$$(I - K) \circ L = L_e \tag{3.30}$$

che ha come soluzione

$$L = (I - K)^{-1} \circ L_e \tag{3.31}$$

L'operatore $(I - K)^{-1}$ è un operatore che agisce da funzione di emissione in quanto trasporta la luce emessa su tutte le superfici. Quindi si può scrivere la soluzione finale come

$$L(x, \omega) = \sum_{i=0}^{n} K^i \circ L_e(x_0, \omega_0) \tag{3.32}$$

e considerando l'n-esimo termine

$$\begin{aligned} L^n(x, \omega) &= K^n \circ L_e(x_0, \omega_0) \\ &= \int_A \dots \int_A L_e(x_0, x_1) G(x_0, x_1) f_r(x_0, x_1, x_2) G(x_1, x_2) \dots \\ &\quad G(x_{n-1}, x_n) f_r(x_{n-1}, x_n, x_{n+1}) dA_{x_0} \dots dA_{x_n} \end{aligned} \tag{3.33}$$

Questo integrale rappresenta una serie di *percorsi* per la luce, ognuno dei quali è caratterizzato da una serie di "rimbalzi" (*bounces*) di lunghezza n. Il fattore $G(x_i, x_{i+1})$ è quello che modella la geometria, mentre A è la superficie su cui si effettua l'integrazione. I percorsi sono specificati da un insieme di vertici; il primo è un punto della sorgente di luce mentre i seguenti sono i punti che il raggio colpisce sulle superfici. Il contributo totale dovuto a tutti i percorsi è dato dall'integrazione su tutte le possibili luci e posizioni della superficie (vedi figura 3.6). Questo significa, come si può evincere dalla formula, effetuare n integrali sulle aree della superficie. Inoltre negli integrali sono presenti funzioni integrande piuttosto difficili (come la f_r, cioè la BRDF del materiale) ed è per questo che viene usato un metodo di Montecarlo. L'algoritmo usato funziona tracciando un raggio da una sorgente in accordo con la sua distribuzione di intensità (assegnando un peso iniziale $weight = 1$) e cercando un punto di intersezione con una superficie. In maniera casuale si decide se assorbire o riflettere il raggio; se viene assorbito si registra il valore del peso alla coordinata (x, y) altrimenti il peso viene modificato per il valore di riflessione delle superficie in accordo con la sua BRDF.

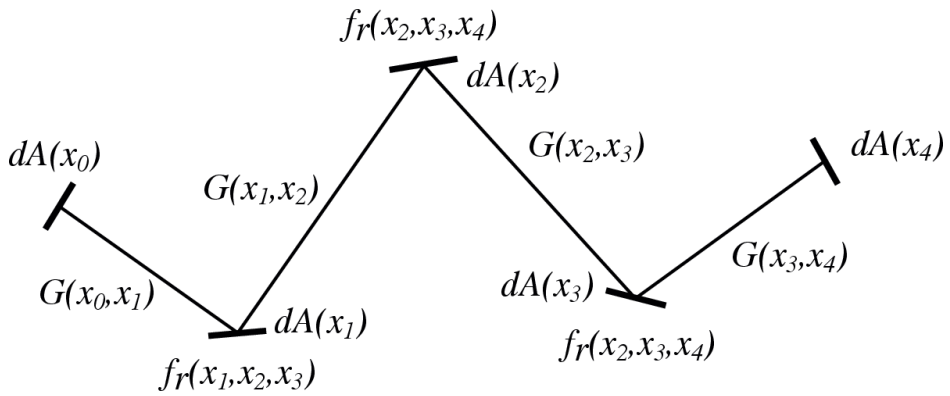


Figura 3.6: *Un possibile percorso di un raggio di luce da un punto x_0 a x_4*

3.2 L'algoritmo per il rendering inverso

In questa sezione verrà descritto l'algoritmo implementato per la stima dell'illuminazione (il programma è stato nominato *Lightrender*). L'input dell'algoritmo è l'immagine di riferimento (una fotografia oppure un rendering sintetico con illuminazione controllata) e il modello tridimensionale corrispondente (in formato .PLY) che rappresenta l'oggetto sul quale si deve stimare l'ambiente di illuminazione. Questi parametri, insieme ad altri, sono memorizzati in un file XML, per la cui descrizione si rimanda all'appendice 5.1.

Dopo aver caricato la fotografia in memoria, il programma inizia a calcolare tramite un software di rendering accelerato in hardware, Gelato^{TM}, i rendering che rappresentano le funzioni di base per l'ambiente di illuminazione. Prima di fare questo è però necessario avere un esatta corrispondenza tra la camera virtuale e quella reale. Infatti per poter confrontare la fotografia reale con i rendering calcolati è necessario che il punto di vista nell'una e negli altri sia il medesimo. Questo allineamento tra la fotografia e la scenda tridimensionale è stato effettuato tramite *TexAlign*, un tool sviluppato al Visual Computing Group del CNR di Pisa [8]. I parametri della camera

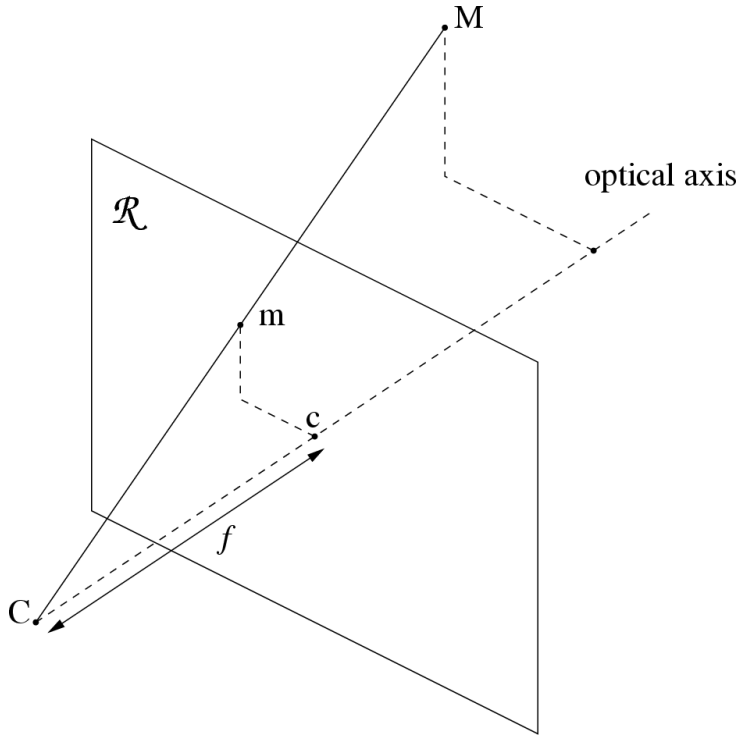


Figura 3.7: *Alcuni dei parametri intrinseci della camera.*

stimata sono quelli *intrinseci* ed *estrinseci*.

I primi sono quelli propri della camera e rimangono gli stessi per ogni foto scattata; essi sono la *lunghezza focale*, ossia la distanza tra il centro di proiezione e la pellicola o il CCD, la risoluzione orizzontale e verticale dell'immagine e il *principal point*, ossia il punto di intersezione tra il centro di proiezione C e il piano dell'immagine. I parametri estrinseci invece descrivono la posizione e l'orientamento della camera rispetto ad un sistema di coordinate e sono definiti da una matrice di rotazione e una di traslazione.

In sintesi, i passi eseguiti dall'algoritmo sono i seguenti:

1. Si generano i rendering corrispondenti alle basi wavelet $\phi\phi$, $\phi\psi$, $\psi\phi$, $\psi\psi$ di livello 0 e si aggiungono all'albero

2. (a) $\forall j = 1, \ldots, m \; \forall k = 1, \ldots, j \; \forall l = 1, \ldots, j$ con m livello di approssimazione scelto, si effettuano i rendering delle basi $\phi\phi^j_{kl}$, $\phi\psi^j_{kl}$, $\psi\phi^j_{kl}$, $\psi\psi^j_{kl}$

   (b) Per ogni rendering effettuato si controlla la sua *qualità*, ossia il numero di pixel illuminati. Se questa è oltre una certa soglia allora nelle successive iterazioni si effettuerà il rendering delle basi di dettaglio più fine, altrimenti verrà scartata.

   (c) Si risolve il sistema lineare

$$c_0 I^0_{0,0} + c_1 I^1_{0,0} + c_1 I^1_{0,0} + \ldots + c_m I^j_{k,l} = P \tag{3.34}$$

   dove c_i sono i coefficienti da stimare, $I^j_{k,l}$ sono i rendering corrispondenti alle basi wavelet e P è la fotografia in input.

Ogni passo di questo algoritmo verrà dettagliato nel resto del capitolo.

3.2.1 Generazione delle Environment Map

A partire da ciascun base wavelet viene creata una corrispondente *environment map* per poter ottenere un rendering che sia illuminato solo da quella base. Una environment map è una particolare tipo di *texture map*, la quale è un immagine che descrive una caratteristica di una superficie tridimensionale (un esempio di texture map è visibile in figura 3.8, dove la caratteristica descritta è il colore del disegno sull'oggetto).

Un environment map è un'immagine che rappresenta l'ambiente di illuminazione che circonda un oggetto. Nel programma sviluppato sono state usate delle *Cubic environment map*, nelle quali l'ambiente è rappresentato sulle sei facce di un cubo ed è quindi memorizzato come 6 texture map quadrate. In figura 3.9 si può vedere come l'environment cube sia memorizzato nella texture.

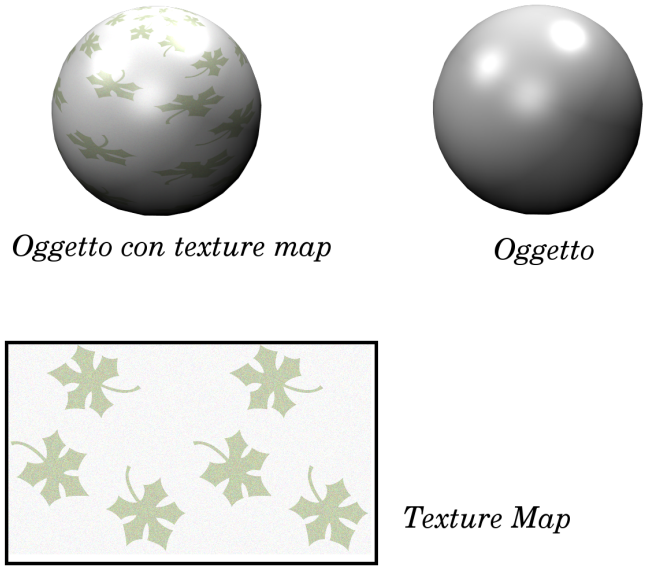


Figura 3.8: *Un esempio di applicazione di texture mapping.*

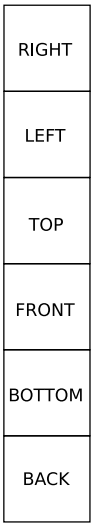
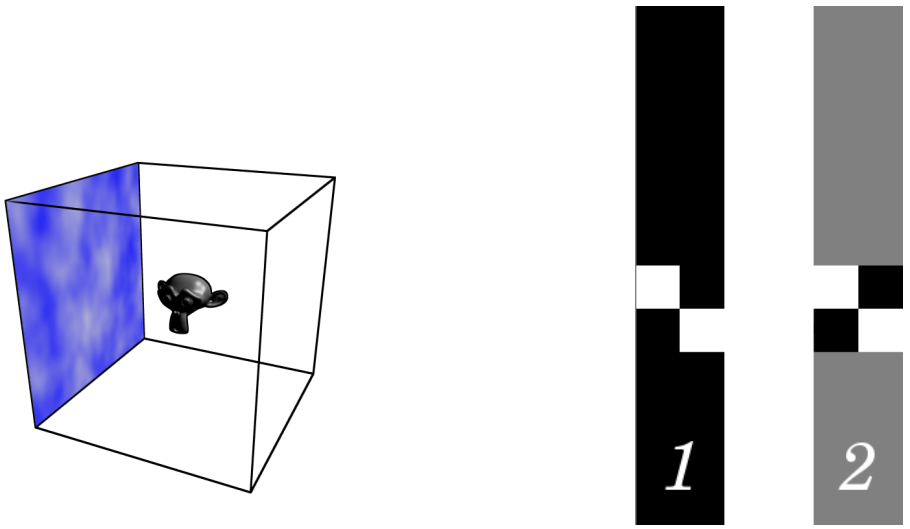


Figura 3.9: *Uno schema di un environment cube.*



(a) Modello illuminato da un environment cube

(b) Una base usata nell'algoritmo

Figura 3.10: *A sinistra un environment cube che illumina un oggetto in mezzo alla scena e destra una base usata (in 1 la base non normalizzata tra* $[-1, 1]$ *e in 2 la stessa ma con valori normalizzati tra* $[0, 1]$*).*

Illuminando l'oggetto con ogni wavelet per ogni faccia del cubo (ottenendo quindi $6 \cdot n$ environment map, con n numero di wavelet), si ottiene un oggetto illuminato da un'"ambiente" composto solo da quella wavelet. Un esempio di environment cube con una texture map in una faccia laterale è mostrato in figura 3.10(a), mentre un environment map utilizzata dall'algoritmo sviluppato è in figura 3.10(b)

Per fare questo è stato usato uno *shader* apposito del software di rendering Gelato. Uno shader (vedi [10]) è un piccolo programma che descrive il comportamento di una luce, una superficie o un volume. La programmabilità degli shader consente di simulare una grande quantità dei comportamenti della fisica della luce. Lo shader usato descrive l'aspetto della luce, in particolare come questa venga emessa da una posizione nello spazio. Esso è

chiamato *envlight* e sfrutta una tecnica chiamata *image based lighting*, introdotta per la prima volta da Paul Debevec [5]. La tecnica consiste nel campionare i pixel dell'environment map e illuminare l'oggetto con il valore di radianza corrispondente ai pixel campionati sulla bitmap. In questa maniera è possibile fare sì che l'environment map appaia proprio come un sorgente di luce presente nella scena. In figura 3.11(a) è possibile vedere un rendering con l'environment map di figura 3.10(b). Sebbene in natura non esistano luci negative, molte environment map hanno intensità negativa poichè la trasformata wavelet decompone l'ambiente di illuminazione in basi che hanno valori negativi, come illustrato in figura 3.3, nella quale le parti nere con segno meno indicano valori minori di zero. La somma di tutte le basi, oppurtunamente pesata con i coefficienti che sono la soluzione dell'algoritmo, darà una environment map con intensità sempre positiva o uguale a zero, simulando correttamente il comportamento naturale della luce.

Di conseguenza le parti visualizzate come nere non corrispondono necessariamente ad assenza di luce (cioè il pixel non è illuminato e vale 0), ma anche all'influenza di una luce *negativa* a causa della quale i pixel possono assumere valori negativi. Per una corretta visualizzazione è quindi necessario normalizzare i valori tra $[0, 1]$ come in figura 3.11(b).

3.2.2 Varianza delle immagini generate

Non tutte i rendering delle basi wavelet contribuiscono a fornire l'illuminazione finale dell'oggetto nello stesso modo. Le wavelet di livello più basso ($j = 0$ o $j = 1$) sono quelle che descrivono l'illuminazione a bassa frequenza, cioè di tipo diffusivo con un dettaglio piuttosto grossolano, mentre quelle di livello più alto (ad esempio per $j = 4$ o $j = 5$) descrivono le alte frequenze, ossia le luci più piccole e direzionali. Questo è dovuto al raffinamento successivo delle wavelet, causato dal fattore di scala sempre più alto all'aumentare del livello

Figura 3.11: *Un rendering con un environment map a valori negativi e lo stesso rendering con valori normalizzati.*

riducendo via via il contributo apportato e limitandolo ad una specifica parte della environment map.

Alcune wavelet invece non danno nessun contributo significativo all'illuminazione dell'oggetto; ad esempio, tutte le wavelet che corrispondono alla faccia dell'environment map che sta *dietro* l'oggetto non ne illumineranno alcuna parte. Di conseguenza aggiungerle alla soluzione del problema porterebbe soltanto ad un suo malcondizionamento. Per poter evitare questo comportamento, per ogni rendering effettuato, si calcola la *varianza* dell'immagine, che definisce un indice di dispersione. Essa è definita come

$$\sigma^2 = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} (p(i,j) - \mu)^2}{MN - 1} \tag{3.35}$$

dove μ è la media dei valori dell'immagine, M e N sono le dimensioni dell'immagine, mentre $p(i,j)$ è il valore del pixel alle coordinate (i,j).

Nel caso in cui la varianza sia troppo bassa significa i valori sono tutti uguali tra loro, poichè si scostano di poco o nulla dalla media, ad esempio se l'immagine è tutta nera. Nei parametri forniti al programma c'è una soglia per la varianza al di sotto della quale un rendering viene scartato.

3.2.3 Calcolo del valore della salienza della fotografia

Un importante contributo del nostro approccio rispetto a quello di Marschner è quello di cercare di ridurre il malcondizionamento del sistema lineare, il quale, quando risolto, ci dà i coefficienti delle basi per ottenere l'ambiente di illuminazione. Infatti nella risoluzione del sistema lineare, se venissero presi in considerazione tutti i pixel la matrice risulterebbe molto sparsa, in quanto molte basi illuminano soltanto alcune porzioni della superificie. Questo perchè una base di livello j alto (che quindi definisce un dettaglio piuttosto fine) associata ad una faccia dell'environment cube risulterà con pochi pixel illuminati dalla parte da cui proviene la luce, cioè dalla faccia

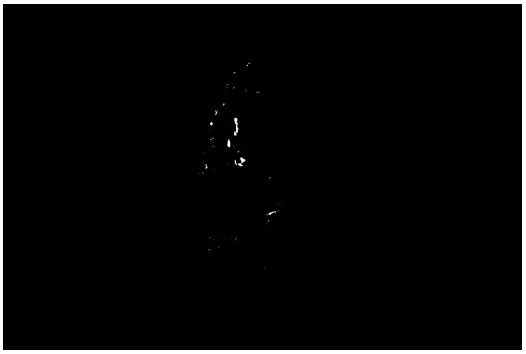


Figura 3.12: *Un rendering di una base che illumina poco l'oggetto*

della base verso l'oggetto. Un esempio di base di livello 0 è visibile in figura 3.11(a) e corrisponde ad una base $\psi\phi$ di livello 0. La stessa base ma al livello 3 è mostrata in figura 3.12 ed ha soltanto poche aree ben illuminate.

Di conseguenza è necessario selezionare solo *alcuni* pixel più significativi della fotografia, detti pixel *salienti*.

Per decidere il grado di signficatività sono state usate delle tecniche di *image processing* sull'immagine, in particolare è stato usato un filtro di *Sobel* il quale è un operatore normalmente utilizzato per rilevare i bordi delle figure presenti in un'immagine. I bordi a cui siamo interessati sono quelli formati dall contrasto tra luce e ombra, infatti l'informazione contenuta in questi punti è quella che ci consente di capire da dove proviene una luce e quindi escludere o ammettere una specifica base.

Come è noto, da [9], il grado di risposta di un operatore che effettua una *derivata* su un'immagine è proporzionale al grado di discontinuità dell'immagine nel punto in cui questo è applicato. In particolare, siamo interessati al comportamento delle differenze (poichè le derivate di funzioni a valori discrete sono definite in termini di differenze) sui valori dei pixel adiacenti dell'immagine. Ad esempio considerando una riga di un'immagine, la differenza tra

due pixel adiacenti è

$$\frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \tag{3.36}$$

Quanto più sarà grande la differenza tra questi pixel tanto maggiore sarà la probabilità di aver rilevato un bordo.

L'operatore di Sobel quindi calcola il valore del gradiente dell'immagine in ogni punto in modo da stimare quanto varia la sua intensità luminosa. Se c'è un brusco cambiamento tra valori di pixel adiancenti significa che quella parte dell'immagine rappresenta un bordo, ad esempio nel passaggio tra una zona d'ombra e una di luce, le quali sono particolarmente interessanti per noi in quanto varieranno da base a base. Matematicamente il gradiente di un'immagine, la quale è una funzione a 2 variabili $f(x,y)$, alle coordinate (x,y), è definito come un vettore

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial x} \end{bmatrix}. \tag{3.37}$$

mentre la sua grandezza è data da

$$\|\nabla \mathbf{f}\| = \sqrt{[{G_x}^2 + {G_y}^2]} \tag{3.38}$$

In pratica l'operatore di Sobel viene calcolato tramite la convoluzione tra l'immagine e un'approssimazione delle derivate; i valori dei kernel di convoluzione sono:

$$\mathbf{K_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \text{e} \quad \mathbf{K_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.39}$$

La grandezza è comparata con un valore di soglia per stimare se un pixel appartiene o meno ad un bordo.

In figura 3.13 si può vedere l'applicazione di un filtro ad un immagine. Tuttavia considerare *solo* i pixel che appartengono ad un edge dell'oggetto

Figura 3.13: *A sinistra un'immagine e destra l'immagine con il filtro di Sobel applicato.*

porterebbe comunque ad un sistema malcondizionato in quanto molte basi hanno quei pixel con valori uguali a zero o prossimi allo zero. L'idea è di considerare solo quelli che sono illuminati dalla maggior parte delle basi. La fotografia è quindi idealmente divisa in una griglia (la dimensione delle celle è un parametro definibile dall'utente) e per ogni cella viene considerato il punto centrale come pixel saliente. Un esempio dei pixel salienti considerati è visualizzato in figura 3.14.

Un'altro semplice accorgimento ha consentito di ottimizzare la scelta dei pixel salienti. L'immagine data in input al programma, specialmente nel caso di fotografie reali, contiene, oltre all'oggetto tridimensionale dato in input al programma, anche un insieme di oggetti di cui non si hanno informazioni e quindi non è possibile tenere in considerazione. Poichè la fotografia e i rendering delle basi sono fatti dallo stesso punto di vista, è stato possibile sovrappore l'*alpha channel* dei rendering sull'immagine. L'alpha channel è un canale aggiuntivo ai canonici R,G,B (che definiscono il colore per le componenti rosso, verde e blu) che viene usato per definire la trasparenza di un'immagine. Esso ha valori che variano nell'intervallo $[0, 1]$, dove 0 significa completamente trasparente e 1 completamente opaco. I rendering hanno il canale dell'alpha channel con valore 1 *soltanto* all'interno della silhouette

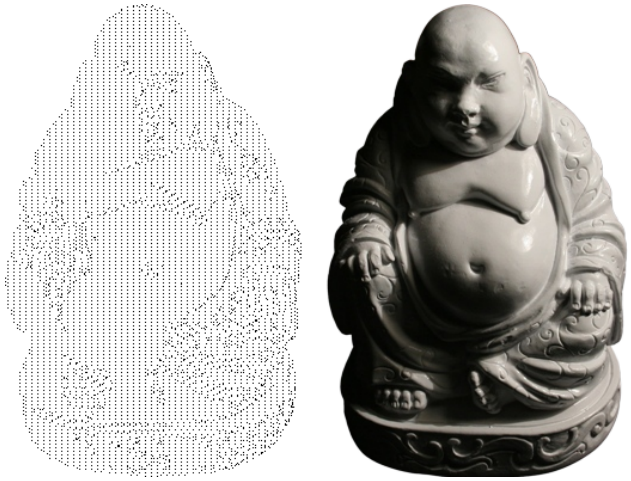


Figura 3.14: *Pixel salienti considerati su una griglia di 200 celle per 200 celle*

dell'oggetto. Sovrapponendo l'alpha channel è stato così possibile eliminare tutti i pixel non influenti nella fotografia, come è possibile vedere in figura 3.15

L'ultimo espediente considerato invece cerca di rimediare all'errore dovuto all'allineamento tra l'inquadratura della fotocamera reale e quella virtuale; insieme con il canale alpha viene anche fatto il rendering di una mappa di profondità dell'oggetto, la quale è un'immagine a scale di grigio in cui è codificato per ogni pixel, a che distanza questo si trovi rispetto alla camera (vedi figura 3.16); più un pixel è bianco più questo è vicino alla camera. Per evitare di campionare pixel non bene allineati lungo l'asse Z (profondità) si scelgono soltanto i pixel che hanno una profondità superiore ad una determinata soglia.

3.2.4 Rendering delle basi

Dopo aver calcolato sia le environment map che i pixel salienti, viene effettuato il calcolo delle basi vere e proprie con un grado di approssimazione

Figura 3.15: *In alto a sinistra la fotografia data in input al programma, a destra l'alpha channel e in basso il risultato della composizione.*

Figura 3.16: *Mappa di profondità di un oggetto.*

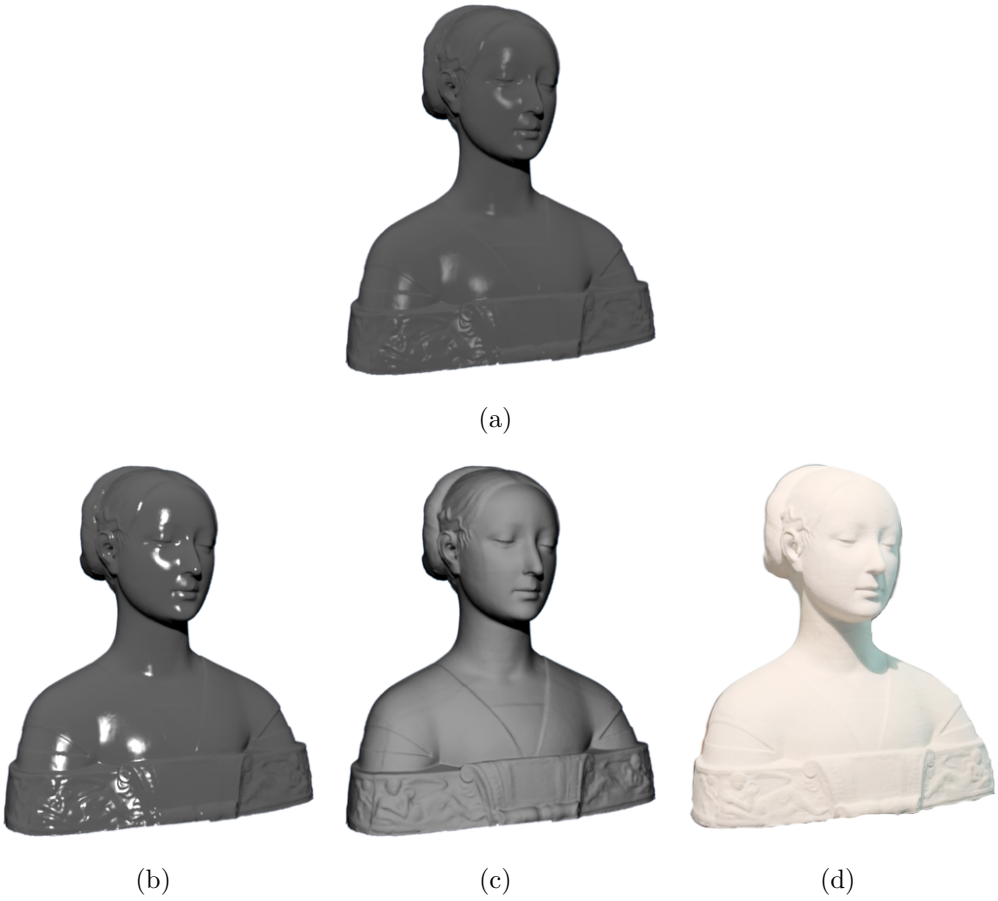
scelto dall'utente tramite un parametro; l'indice j delle wavelet va da 0 al livello scelto.

Come accennato precedentemente, è stato fatto uso di un software di rendering accelerato in hardware, Gelato, il quale ha consentito di diminuire i tempi di calcolo, poichè il numero delle basi cresce esponenzialmente con l'aumentare del livello j (e quindi del dettaglio da loro rappresentato) delle wavelet. Questo software sfrutta oltre alla CPU di sistema anche la GPU (*Graphics Processing Unit*) della scheda grafica per velocizzare i calcoli.

Per calcolare in maniera corretta l'equazione di rendering [12],viene usato il *Montecarlo raytracing*, descritto in 3.1.3.

La BRDF usata è quella di Phong [18], che ha come parametri il grado di specularità e di diffusione dell'oggetto, o quella di Oren-Nayar [15], che è più adatta per simulare materiali di creta ed ha come parametri il grado di diffusione e l'irregolarità della superficie; la BRDF è stata stimata manualmente tramite alcuni rendering di prova. In figura sono illustrate tre passi della stima della BRDF di una scultura di un busto di donna, per la quale si è usata una BRDF di Oren-Nayer.

Come per le luce e le environment map descritte in 3.2.1, anche in questo caso il software mette a disposizione uno *shader* che restituisce il colore di un punto alla superficie, consentendo di simulare ogni tipo di BRDF a partire da parametri quali la normale al punto, la posizione della camera, il grado di specularità e diffusione di un materiale e altri parametri. La scelta della BRDF è un punto delicato dell'algoritmo e deve essere effettuata molto attentamente, infatti per un caso di prova (vedi 4.4) un'impostazione sbagliata della BRDF aumenta l'errore di stima dell'11%.



(a)

(b) (c) (d)

Figura 3.17: *Un esempio di tre passi effettuati per la stima della BRDF: in 3.17(a) la specularità è troppo alta e la superficie appare troppo piatta, in 3.17(b) la specularità è sempre troppo alta, mentre in 3.17(c) la BRDF è simile a quella di riferimento 3.17(d), anche se non è equivalente.*

3.2.5 Risoluzione del sistema lineare

Una volta che le basi di un livello j sono state renderizzate per ogni faccia dell'environment cube viene risolto il sistema lineare del tipo $Ax = b$ che consente di ottenere una stima approssimata dell'ambiente di illuminazione. Questo sistema è composto da una matrice A di dimensioni $m \times n$, dove m è il numero dei pixel salienti mentre n è il numero delle wavelet considerate, b è un vettore di m elementi che contiene i valori dei pixel salienti campionati nella foto data in input al programma. L'obiettivo è di ottenere un sistema come in 2.9 del tipo

$$\begin{bmatrix} I_{00}(x,y) & I_{01}(x,y) & \dots & I_{0n}(x,y) \\ I_{10}(x,y) & I_{11}(x,y) & \dots & I_{1n}(x,y) \\ \vdots & & & \\ I_{m0}(x,y) & I_{m1}(x,y) & \dots & I_{mn}(x,y) \end{bmatrix} \begin{bmatrix} \vdots \\ x_j \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ b_i \\ \vdots \end{bmatrix} \tag{3.40}$$

I valori di $I_{ij}(x,y)$ è il valore del i-esimo pixel saliente di coordinate (x,y) che appartiene alla base j (in questo caso j non ha a che fare con il livello delle wavelet). Come nel caso di Marschner, la colonna di indice j conterrà i valori dei pixel del modello tridimensionale che viene illuminato dalla base j-esima. La risoluzione del sistema permette di assegnare un peso ad ognuna di queste basi in modo che la loro somma sia equivalente all'ambiente di illuminazione della foto. Tuttavia, come già anticipato, il problema è spesso mal condizionato e risolverlo tramite metodi come Gauss-Jordan o Jacobi può portare ad una soluzione errata.

Per questo è stato adottato il metodo della decomposizione a valori singolari (SVD, *Singular Value Decomosition*). Il metodo della SVD decompone una matrice $A \in \mathbb{C}^{m \times n}$ in tre matrici

$$A = U \Sigma V^H \tag{3.41}$$

dove $U \in \mathbb{C}^{m \times m}$ è una matrice unitaria, $V \in \mathbb{C}^{n \times n}$ è anch'essa unitaria e

$\Sigma \in \mathbb{R}^{m \times n}$ è una matrice diagonale con elementi $\sigma_{ii} = \sigma_i$ tali che

$$\sigma_1 \geqslant \sigma_2 \ldots \geqslant \sigma_p \geqslant 0, \quad p = \min\{\text{m,n}\} \tag{3.42}$$

La matrice Σ è univocamente determinata, mentre non necessariamente lo sono U e V.

Essendo U e V matrici unitarie, quindi con le matrici inverse uguali alle matrici trasposte, e Σ singolare, con matrice inversa uguale alla stessa matrice ma con elementi che sono i reproci, si può ricavare l'inversa di A come

$$A^{-1} = V\Sigma^{-1}U^T \tag{3.43}$$

e risolvere di conseguenza il sistema lineare. L'algoritmo per il calcolo della SVD utilizzato è tratto da[19].

Dopo aver calcolato i "pesi", ossia le incognite x_j del sistema lineare, queste sono moltiplicate per i rendering delle rispettive basi

$$p_{app} = \sum_{i=1}^{k} I_i x_i \tag{3.44}$$

dove p_{app} è l'approssimazione raggiunta da quel livello di dettaglio delle wavelet ed è data dalla somma di tutti gli I_i opportunamente pesati.

Due ulteriori fattori vengono presi in considerazione per rendere più stabile il sistema.

- Quando si costruisce la matrice A con i pixel salienti delle basi, se ogni base ha il valore del pixel uguale a zero, allora quel pixel viene scartato poichè significa che nessuna delle basi considerate lo illumina e risulterebee in una riga della matrice di soli zeri.

- Se una base è scarsamente illuminata [3] allora saranno scartate le successive sottbasi, in altre parole quelle con indice j maggiore e quindi con

[3]questo conto viene fatto semplicemente contando i pixel illuminati, cioè diversi da zero, divisi per i pixel totali, sempre limitati all'interno dell'alpha channel

area di illuminazione ancora più limitata e definita, vengono ignorate nelle successive iterazioni.

3.2.6 Calcolo dell'errore e generazione dell'ambiente di illuminazione finale

Una volta che il sistema lineare è stato risolto, viene calcolato l'errore in percentuale commesso. Siano $I_1(x, y), \ldots, I_k(x, y)$ i rendering delle k basi wavelet considerate nei passi precedenti, la SVD permette di calcolare i coefficienti che andranno a moltiplicare i rendering delle basi per poter ottenere la stima p_{app}. All'interno dell'alpha channel del modello illuminato viene quindi fatta la differenza in valore assoluto dell'immagine stimata con l'immagine originale, cioè

$$Err = \frac{p_{app} - p}{M} \tag{3.45}$$

dove M sono i soli pixel presenti nell'alpha channel.

L'ambiente di illuminazione viene quindi calcolato moltiplicando ogni base wavelet (come quelle mostrate in figure 3.3, considerando basi distinte per facce dell'environment cube distinte) per il relativo coefficiente dato dalla risoluzione del sistema lineare. Più precisamentte, siano $b_1, \ldots, b_k$ le basi wavelet per le quali i sono stati effettuati i rendering, e siano $x_1, \ldots, x_k$ i coefficienti stimati, allora l'ambiente di illuminazione è dato da

$$Env = \sum_{i=1}^{k} b_i x_i. \tag{3.46}$$

In pratica, nel codice questo corrisponde a sommare tra loro le immagini che rappresentano le wavelet, ognuna moltiplicata per il relativo coefficiente.

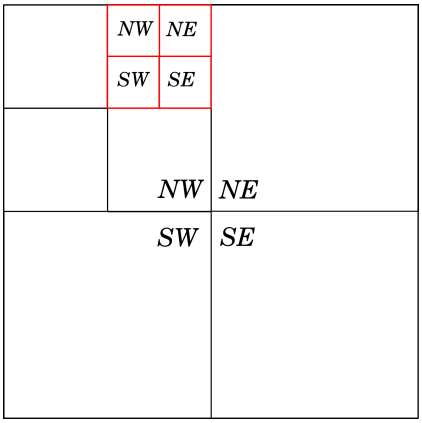


Figura 3.18: *Quadtree*

3.2.7 Accesso alla struttura dati

La struttura dati che contiene i rendering che costituiscono la basi algebriche per poter ottenere la stima dell'illuminazione è una variante di un *quadtree*, il quale è un albero che ha esattamente quattro figli per ogni nodo. Esso viene usato normalmente per partizionare una spazio bidimensionale in quattro regioni in maniera ricorsiva. Come si può vedere in figura 3.18 il quadrante in alto a sinistra è identificato con *NW*, ossia nord ovest, quello in alto a destra è *NE*, nord est, mentre quelli in basso sono *SW*, sud ovest, e *SE*, sud est. Nella figura 3.18 si può vedere come il quadrante NW sia stato partizionato in quattro quadranti, uno dei quali (il NE) è stato ulteriormente partizionato. Si può vedere un quadrante a livello k come un vettore di identificativi (scelta tra NW, NE, SE, SW) di lunghezza k. Ad esempio il quadrante SE più piccolo (al livello 3) della figura 3.18 si può vedere come (NW, NE, SE) poichè NW rappresenta il primo quadrante in alto a sinistra, il secondo elemento NE è indicato in rosso nella figura e rappresenta il livello più in dettaglio del precedente e infine il terzo elemento SE rappresenta l'elemento a cui vogliamo accedere. Il quadtree è la struttura dati ideale per

rappresentare l'informazione data dalle wavelet, poichè come si può vedere in figura 3.3 ogni wavelet di livello $j+1$ suddivide in 4 parti quella del livello precedente. In realtà , poichè è necessario avere a disposizione tre wavelet diverse ($\phi\psi$, $\psi\phi$, $\psi\psi$) più una singola *scaling function* $\phi\phi_{0,0}^{0}$ è stata creato un albero che contiene tre quadtree differenti per ogni wavelet piu la singola scaling function come radice. Per avere un accesso più veloce e diretto ad ogni wavelet si è scelto di memorizzare l'albero in un vettore. Di conseguenza la formula per accedere ad un elemento generico del quadtree è

$$(4^p f) - 1 + \sum_{i=0}^{p-1} pos[p-i]4^i \tag{3.47}$$

dove *pos* è un vettore lungo p che contiene le regioni a cui vogliamo accedere scendendo in profondità nel quadtree, f è il tipo di wavelet (ad ogni funzione è associato un intero, $\phi\phi = 0$, $\phi\psi = 1$, $\psi\phi = 2$, $\psi\psi = 3$). Nel caso si voglia accedere al livello 0 viene restituito direttamente l'intero associato alla funzione f. Dato che vogliamo un intero per indicizzare un vettore, i quadranti sono associati a degli indici, in particolare $NW = 1$, $NE = 2$, $SW = 3$, $SE = 4$ che vengono memorizzati nel vettore. Gli indici delle wavelet vengono trasformati da una funzione *quadTreePosition* che, presi in ingresso gli indici j, k, l che definiscono una wavelet, restituisce un vettore di posizioni. Ad esempio se $j = 2, k = 1, l = 0$, essa resitituisce il vettore dell'esempio precedente, ossia (NW, NE, SE). Il termine $(4^p f) - 1$ è necessario per accedere alla wavelet voluta, mentre la sommatoria consente di scendere nell'albero. Riprendendo l'esempio di prima il quadrante SE più piccolo della funzione $\psi\phi$ (che vale 2) avrà come vettore $p = (1, 2, 4)$, mentre l'indice per accedere al vettore sarà $(4^3 \cdot 2) - 1 + 4 + 2 \cdot 4 + 1 \cdot 16 = 119$. Inoltre per ogni funzione wavelet è necessario avere 6 alberi, uno per ogni faccia dell'environment cube. In figura 3.19 è mostrato un esempio della struttura dati per i livelli 0 e 1 per una faccia dell'environment cube.

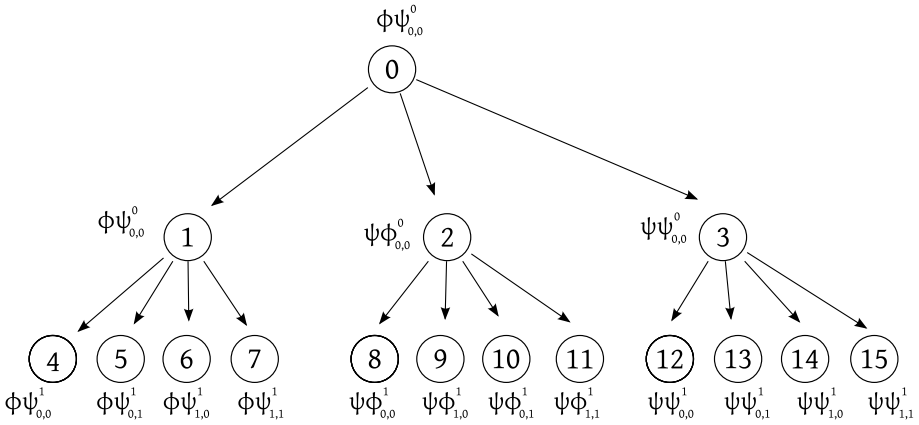


Figura 3.19: *La struttura dati ad albero usata per memorizzare le wavelet con livelli 0 e 1.*

3.2.8 Complessità computazionale

La parte principale dell'algoritmo è il ciclo in cui si calcolano i rendering delle basi wavelets e in cui si risolve il sistema lineare. Il numero delle basi wavelet

Algorithm 1 Ciclo principale dell'algoritmo

Effettua i rendering delle basi di livello 0
for all wavelet $\in \{\phi\psi^j_{k,l}, \psi\phi^j_{k,l}, \psi\psi^j_{k,l}\}$ **do**
  for all facce s dell'environment cube **do**
    if il rendering della base wavelet corrispondente agli indici $j-1, k, l$ è presente nell'albero **then**
      Calcola il rendering della base wavelet corrispondente agli indici j, k, l e faccia s.
    end if
    if il rendering supera la soglia della varianza **then**
      Aggiungi all'albero un nodo con il rendering effettuato.
    end if
  end for
end for
Risolvi il sistema lineare $Ax = b$

dipende dal livello di dettaglio m scelto; in tutto esse sono $3 \cdot 6 \cdot \sum_{j=0}^{m} 4^j$ basi, poichè 3 sono le basi wavelet e 6 sono le facce dell'environment cube. Considerando che $\sum_{j=0}^{m} 4^j$ è una progressione geometrica e che $\sum_{j=0}^{m} x^j = \frac{1-x^{m+1}}{1-x}$, allora il numero delle basi è $O(\frac{4^{m+1}-1}{3})$. I rendering che devono essere generati, sono, nel caso peggiore, $O(\frac{4^{m+1}-1}{3})$. Come esempio, consideriamo nella figura 3.3 solo la base $\phi\psi$: il livello di dettaglio è $m = 1$ e il totale delle basi di tipo $\phi\psi$ è 5.

Tuttavia bisogna considerare che, pur essendo una complessità esponenziale nel caso peggiore, nel caso medio molte di queste basi non vengono

renderizzate, in quanto se al livello di dettaglio $j-1$ una base non contribuisce significativamente all'illuminazione (ossia se è quasi tutta nera e la sua varianza è molto bassa) allora non viene aggiunta all'albero e quindi anche tutte le sottobasi non vengono nè generate nè renderizzate. Inoltre raramente si ha alcun contributo di illuminazione quando si considerano basi con dettaglio maggiore di $m = 5$, come è possibile vedere in figura 3.12, per cui l'algoritmo effettua il rendering solo di un sottoinsieme abbstazna limitato delle basi necessarie. Nei test pratici, al livello $j = 6$ sono stati effettuate poche centinaia di rendering. Il passo del rendering dipende da come è implementato il software di rendering, dal numero di facce dell'oggetto e da altri parametri. Come mostrato da [22], la complessità di un rendering che implementa il Montecarlo raytracing varia da $O(N^2)$ a $O(N \log N)$, dove N è il numero di patch in cui è suddivisa la geometria della scena. Praticamente la maggiorparte del tempo impiegato è in questo passo e nei test effettuati il tempo impiegato varia da pochi minuti a circa dieci minuti per ogni rendering sulla macchina usata nei test. Il calcolo della varianza è invece $O(MN)$ dove M e N sono le dimensioni dell'immagine. Infine, la risoluzione del sistema lineare viene fatta tramite SVD il quale ha una complessità nell'implementazione scelta $O(min(nm^2, mn^2))$ dove n è il numero di righe e m è il numero di colonne (nel nostro caso rispettivamente pixel salienti e basi wavelet).

Capitolo 4

Risultati

In questo capitolo vengono mostrati i risultati ottenuti tramite l'algoritmo sviluppato. Gli esempi mostrati riguardano la stima di un ambiente di illuminazione di :

- Una scena reale con un statua di un Budda, il cui modello geometrico tridimensionale è liberamente disponibile in rete nel repository di aimatshape.net, mentre la fotografia è stata scattata con una luce situata a destra dell'oggetto con una macchina fotografica Canon EOS 350D.
- Una scena sintetica con un modello di una statua di un *gargoyle* (anch'esso disponibile nel repository aimatshape.net). La "fotografia" è stata calcolata tramite un software che risolve con il metodo Montecarlo l'equazione di rendering 2.1, in modo da simulare una fotografia reale.
- Una scena reale con un busto di donna dello scultore rinascimentale Francesco Laurana.

Un ulteriore test è stato effettuato su una scena sintetica della statua del Budda, calcolata similmente a quella del gargoyle.

Nel caso dei test sintetici sono state provate più condizioni di illuminazione, ad esempio con una luce laterale, due dall'alto oppure con tre fari che illuminano da dietro, da sotto e da un lato.

Prima di esporre i risultati è però necessario fare alcune considerazioni: il software usato, NVIDIA Gelato, calcola una soluzione approssimata ma veloce dell'equazione di rendering, cercando di sfruttare il più possibile le risorse di calcolo presenti, in partcolare la GPU oltre che la CPU. Questo consente di accelerare i tempi necessari per la risoluzione del problema, in genere abbastanza lunghi. Nel PC usato per effettuare i test, un Athlon 3500+ con scheda grafica Geforce 6600GT in media i tempi sono di circa 10 minuti[1] a rendering per ogni lato dell'environment cube e per ogni base; considerando che solitamente è necessario avere a disposizione qualche centinaio di wavelet per approssimare correttamente la soluzione, il tempo impiegato è di circa un giorno o due di calcolo. Usando un software di rendering più preciso, come ad esempio *Physically Based Rendering* [17] , probabilmente anche la soluzione sarebbe stata più precisa, tuttavia i tempi di calcolo sarebbero aumentati notevolmente.

Il software scritto inoltre scala verso l'alto con il diminuire dei tempi di calcolo dei motori di rendering: usando ad esempio i metodi illustrati in [16] un rendering, anche in questo caso con molte approssimazioni e non completamente corretto dal punto di vista fisico, viene calcolato in poche decine di secondi. Le assunzioni fatte sono le stesse di Marschner e Sato come discusso in 2.2 e 2.3 con in più la limitazione sugli oggetti, i quali devono essere opcahi non colorati illuminati da luce bianca.

[1] con un oggetto di 500.000 triangoli

4.1 Modello di un Budda con fotografia reale

Il primo test è stato eseguito con una statuetta di gesso che rappresenta un Budda (il cui modello è composto da circa 500.000 triangoli) illuminata con una luce proveniente da destra tramite una comune lampada da scrivania. In figura 4.1(a) è mostrata la fotografia data in input al programma, mentre in figura 4.1(b) è possibile vedere il risultato del calcolo. Il calcolo è stato effettuato con 389 basi wavelet su 6144 totali. Le restanti basi o non sono state calcolate oppure non sono state contate nel calcolo della soluzione in quanto non contribuivano in maniera sostanziale all'illuminazione della scena. I pixel salienti sono stati scelti suddividendo l'immagine in una griglia di 200×200 celle e campionando in base al loro grado di significatività, come discusso in 3.2.3. La BRDF è stata stimata manualmente tramite 3 iterazioni confrontando un rendering del modello tridimensionale con la fotografia reale.

La tabella 4.1 riassume il risultato: E è l'errore percentuale, cioè la differenza in valore assoluto tra l'immagine di riferimento e quella stimata. L'ambiente di illuminazione stimato, rappresentato come un environment cube map come discusso in 3.2.1, è visibile in figura 4.2. Una rappresentazione dei pixel salienti è mostrata in figura 4.3.

Tabella 4.1: *Statua del Budda, fotografia reale*

Basi totali	Basi Effettive	E	Pixel salienti
6144	389	9%	5263

I tempi di calcolo sono stati di circa 8 minuti per ogni rendering, quindi in totale sono necessari circa 2 giorni di precalcolo per avere i rendering di tutte le 389 basi usate. L'errore è dovuto in parte al calcolo manuale e approssimato della BRDF, differente tra i rendering delle basi e quella dell'oggetto

(a) Originale

(b) Reilluminato

Figura 4.1: *Confronto tra foto reale e rendering con ambiente stimato:4.1(a) rappresenta la foto dell'oggetto reale mentre 4.1(b) la foto dell'oggetto reilluminato. In figura 4.4 invece è mostrata la differenza tra le due precedenti immagini.*

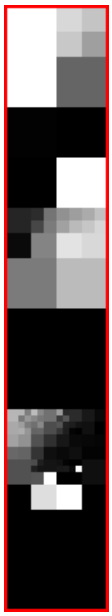


Figura 4.2: *L'ambiente di illuminazione stimato per la fotografia della statua reale del Budda.*

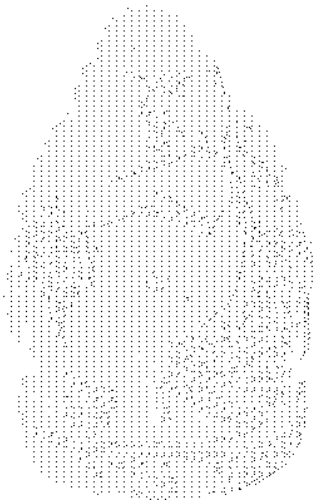


Figura 4.3: *I pixel salienti usati per la stima della scena reale con il Budda.*

Figura 4.4: *Differenza tra l'immagine reale e l'immagine con l'ambiente di illuminazione stimato*

reale, in parte al non corretto allineamento tra la camera virtuale e la camera reale e al numero di basi utilizzate, poichè i riflessi speculari sono componenti ad alta frequenza dell'ambiente di illuminazione e vengono rappresentati efficacemente solo da wavelet di livello alto. L'errore è di circa il 9%, in figura 4.4 è mostratata una sua rappresentazione ottenuta come differenza tra la fotografia di input e un rendering con l'ambiente di illuminazione stimato.

4.1.1 Considerazioni

Come si può vedere anche visivamente, la stima non è perfetta; questo è dovuto ai vari errori discussi nel paragrafo precedente. Tuttavia si può vedere come il modello reilluminato riceva correttamente la luce dalla parte destra della scena e invece sia oscurato nella parte sinistra, come avviene nella fotografia reale dato che l'oggetto è illuminato principalmente da una lampada da destra (oltre ovviamente all'illuminazione indiretta che proviene dalle altre superfici).

4.2 Modello di un Budda con fotografia sintetica

In questo caso la fotografia di input non è reale, ma è il risultato di un rendering effettuato con un software che simula l'equazione di rendering. Il modello geometrico usato per effettuare la fotografia "sintetica" è lo stesso dei rendering delle basi wavelets. In questo caso, dato che l'ambiente poteva essere agevolmente controllato essendo virtuale, sono state provate due configurazioni: nella prima una luce illumina dall'alto il modello, nella seconda le luci sono sia dietro che di lato rispetto all'oggetto. Il tempo di calcolo è esattamente uguale al caso della fotografia reale, in quanto la posizione della camera che inquadra la fotografia sintetica è la stessa della camera del test precedente; in questo modo è stato possibile evitare il precalcolo di tutti i rendering delle basi wavelets, poichè era già stato effettuato. L'unica differenza rispetto al caso precedente, oltre alla fotografia in input, sono i punti salienti campionati sulla fotografia sintetica, che sono stati scelti effettuando un campionamento su una griglia di 200×200 celle. La BRDF è più verosimile rispetto al caso precedente, in quanto i suoi valori nei dei rendering delle basi wavelet e nella rendering sintetico sono gli stessi. In figura 4.5(a) è mostrata la fotografia sintetica del primo setup, con una luce posta sopra la statua, mentre in figura 4.5(b) si vede il modello reilluminato tramite l'ambiente di illuminazione stimato.

In figura 4.6(a) è mostrata la fotografia sintetica del secondo setup, con più luci poste intorno al modello, mentre il relighting è mostrato in 4.6(b).

In tabella 4.2 sono riassunti i risultati: In figura 4.8(a) è rappresentato l'ambiente di illuminazione stimato per il primo setup, mentre in figura 4.7 è illustrato il campionamento dei pixel salienti.

In questo caso l'errore è minore del precedente, è circa del 4%, poichè non ci sono problemi di allinemento delle camere (essendo entrambe virtuali) e

(a) Originale

(b) Reilluminato

Figura 4.5: *Primo setup, luce in alto: 4.5(a) rappresenta la foto dell'oggetto sintetico mentre 4.5(b) la foto dell'oggetto reilluminato.*

(a) Originale

(b) Reilluminato

Figura 4.6: *Secondo setup, varie luci intorno all'oggetto: 4.6(a) rappresenta la foto dell'oggetto sintetico mentre 4.6(b) mostra la foto dell'oggetto reilluminato.*

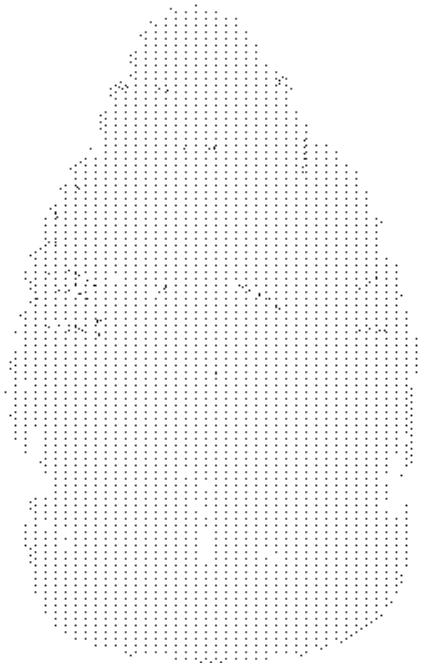
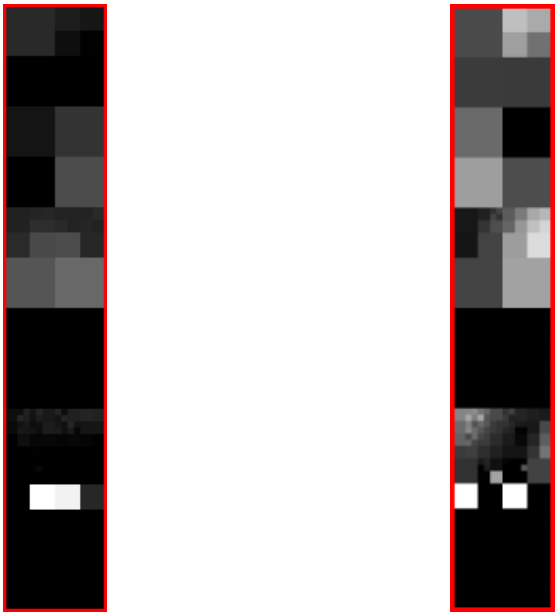


Figura 4.7: *I pixel salienti usati per la stima della scena sintetica con il Budda.*



(a) Ambiente con luce da sopra

(b) Ambiente con varie luci

Figura 4.8: *L'ambiente di illuminazione stimato per la fotografia sintetica della statua del Budda: in figura 4.8(a) nel caso di illuminazione dall'alto e in figura 4.8(b) con più luci poste intorno al modello.*

Tabella 4.2: *Statua del Budda, fotografia sintetica*

Luci	Basi totali	Basi Effettive	E	Pixel salienti
Alto	6144	105	3.5%	5263
Varie	6144	115	6.4%	4012

della stima manuale della BRDF. In figura 4.9(a) e 4.9(b) sono mostrate le differenze tra le immagini.

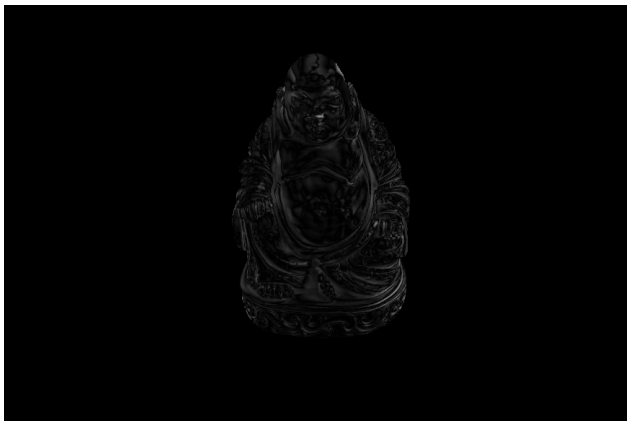
Per testare l'algoritmo in un altro caso sintetico, abbiamo inoltre provato un caso in cui l'ambiente di illuminazione non era una semplice luce direzionale che proveniva da un punto soltanto ma una environment map (prelevata dal sito di Paul Debevec, vedi [6]), la quale è un HDRI (vedi 3.1.2) che descrive l'illiminazione dell'interno di una cattedrale e contiene molte luci da ogni angolazione, tranne che dal basso, con un alto contrasto tra le parti chiare e quelle scure. Le figure 4.10(a) e 4.10(b) rappresentano rispettivamente il rendering con l'ambiente di illuminazione originale (il quale è mostrato in 4.11) e quello con l'ambiente di illuminazione stimato dall'algoritmo.

Come si può vedere l'illuminazione stimata è prossima a quella reale, in particolare nella fronte del modello e nella pancia, considerando che ci sono solo 86 basi wavelets e 6286 pixel salienti.

4.2.1 Considerazioni

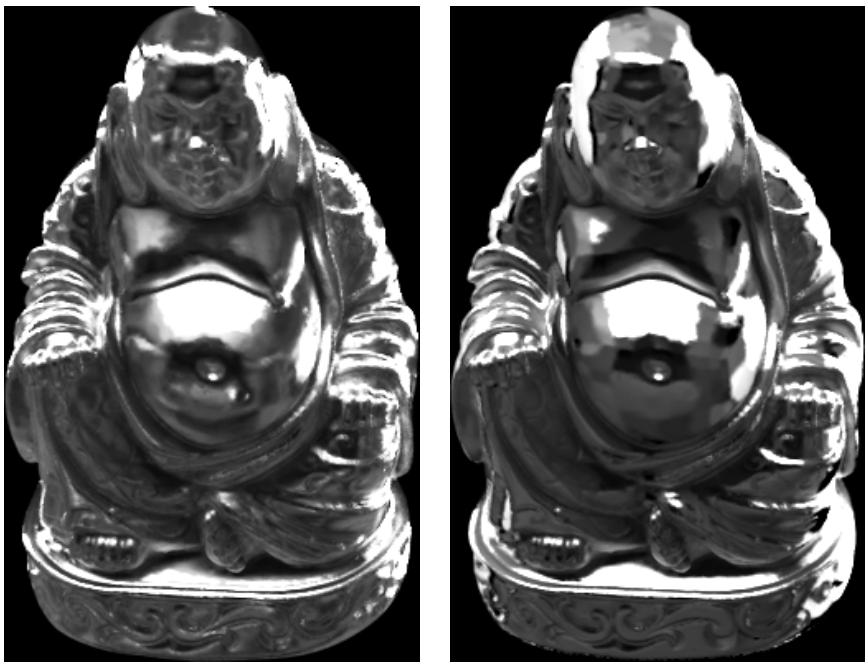
In questo caso le considerazioni sono simili a quelle della scena reale, la direzione della luce viene correttamente stimata anche se in questo caso la componente speculare dela BRDF del modello è decisamente più attenuata, cosa che potrebbe indurre un malcondizionamento del problema come discusso in [20]. Nel caso di più luci la stima è meno corretta e l'errore aumenta di qualche punto percentuale.

(a) Differenza tra rendering originale e stimato con luce dall'alto



(b) Differenza tra rendering originale e stimato con varie luci

Figura 4.9: *Differenze tra le immagini "reali" e quelle stimate: in 4.9(a) nel caso di una luce dall'alto, mentre in 4.9(b) nel caso di varie luci intorno al modello.*



(a) Originale (b) Reilluminato

Figura 4.10: *Rendering con ambiente di illuminazione di una cattedrale: 4.10(a) con ambiente di illuminazione originale, 4.10(b) con ambiente di illuminazione stimato dall'algoritmo.*

Figura 4.11: *L'ambiente di illuminazione usato per il rendering di 4.10(a).*

4.3 Modello di un Gargolye con fotografia sintetica

Anche in questo caso la foto di input è sintetica, calcolata coma il precedente test, ma rappresenta la statua di un *gargoyle*, il cui modello geometrico è di circa 300.000 poligoni. Abbiamo provato due illuminazioni differenti: nel primo caso una luce è situata dietro la camera, nelll'altro la luce si trova di lato a sinistra rispetto alla camera. Il rendering di ogni singola base ha richiesto circa 4 min, mentre le basi usate sono stati circa 150 (sempre su 6144 dato che il livello di dettaglio è stato scelto come nel caso precedente), quindi in tutto il tempo di precalcolo è stato di 10 ore circa.

Così come nei casi precedenti, la dimensione della griglia di campionamento per i pixel salienti è stata di 200×200 celle, mentre la BRDF è simile ma non equivalente tra i rendering delle basi wavelets e la fotografia sintetica, poichè sono stati usati due differenti software per effettuare i rendering per le basi e la fotografia. Nella tabella 4.3 sono riassunti i valori, mentre in figura 4.12(a) e 4.12(a) sono mostrati rispettivamente la fotografia sintetica e il modello reilluminato tramite l'ambiente di illuminazione stimato.

Tabella 4.3: *Statua Gargoyle*

Basi totali	Basi Effettive	E	Pixel salienti
6144	71	6%	5263

In figura 4.13 è rappresentato l'ambiente di illuminazione stimato per il primo setup, mentre in figura 4.14 è illustrato il campionamento dei pixel salienti.

(a) Originale

(b) Reilluminato

Figura 4.12: *Un modello di un gargoyle illuminato di fronte: 4.12(a) rappresenta la foto dell'oggetto sintetico mentre 4.12(b) mostra la foto dell'oggetto reilluminato.*

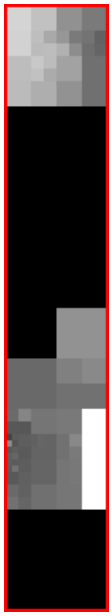


Figura 4.13: *L'ambiente di illuminazione stimato per la fotografia "sintetica" del modello del gargoyle.*

Figura 4.14: *I pixel salienti usati per la stima della scena del gargoyle.*

4.3.1 Considerazioni

Così come nei test precendenti, viene catturata la direzione della luce, anche se l'ombra sull'ala della statua non viene stimata correttamente; questo probabilmente è dovuto alla soglia del filtro di Sobel, in quanto la differenza tra luce e ombra e in quel punto è minima e la grandezza del gradiente non va oltre la soglia.

4.4 Modello di un busto di donna con fotografia reale

Il terzo test riguarda la stima dell'ambiente di illuminazione di una scena reale che comprende una statua di gesso la quale rappresenta un busto di donna. Sono stati provati tre differenti ambienti di illuminazione:

1. Una lampada da scrivania (la stessa usata nel primo test con la stauta del Budda) illumina da sinistra l'oggetto.

2. Due luci al neon della stanza in cui sono state scattate le foto illuminano da dietro e da davanti la statua.

3. Nessuna illuminazione artificiale presente ma solo l'illuminazione naturale dal lato destro che proviene dalle finestre.

Il modello geometrico della statua è composto da 200.000 triangoli, mentre il rendering di ogni wavelet è stato di circa tre minuti e mezzo. Le basi calcolate sono state 186 su un totale di 384 (poichè l'approssimazione scelta si ferma al livello $j = 2$), per un totale di circa 10 ore di calcolo. Le basi effettivamente usate però sono soltanto 75 poichè molte basi non sono state contate nella soluzione del sistema lineare, in quanto davano un contributo molto limitato all'illuminazione.

Tabella 4.4: *Statua del busto di donna*

Luci	Basi totali	Basi Effettive	E	Pixel salienti
Sinistra	384	75	7.5%	6136
Neon dietro e davanti	384	75	7 %	6142
Naturale	384	75	5%	6047

Come nel prima caso, in cui si aveva a che fare con la fotografia di un modello reale, la BRDF è stata stimata tramite alcune iterazioni. La dimensione della griglia di campionamento è stata di 200×200 celle, per un totale di 6136 pixel salienti usati. In figura 4.15(a) e 4.15(b) sono mostrati sia l'input che l'output del programma, mentre l'ambiente di illuminazione stimato è in figura 4.16.

In tabella sono riassunti i dati

In figura 4.17 e 4.18 sono mostrati i pixel salienti visualizzati e una differenza tra la foto reale e quella reilluminata.

In figura 4.19(a) e 4.19(b) è mostrato il caso con le luci al neon da dietro e da davanti, mentre 4.20(a) e 4.20(b) illustra il caso con solo l'illuminazione naturale.

4.4.1 Considerazioni

In questo caso la BRDF ha inciso molto sulla soluzione del problema. Infatti il materiale della statua fotografata è il gesso, che viene reso male con una BRDF di Phong mentre è più adatta una BRDF come quella di Oren-Nayar. Passando dall'una all'altra l'errore è sceso di parecchi punti percentuale. Tuttavia anche in questo caso si è resa necessaria una stima manuale che non può coincidere perfettamente con quella del modello reale, anche perchè il materiale reale è traslucente e questo fenomeno non viene modellato affatto dalla BRDF di Oren-Nayar, ma bisognerebbe tenere in considerazione una

(a) Originale

(b) Reilluminato

Figura 4.15: *Un statua che rappresenta un busto di donna: 4.15(a) è la foto dell'oggetto mentre 4.15(b) mostra la foto dell'oggetto reilluminato.*

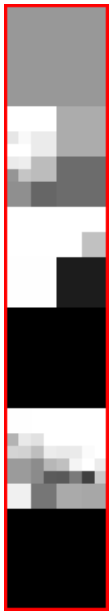


Figura 4.16: *L'ambiente di illuminazione stimato per foto della statua del busto di donna.*

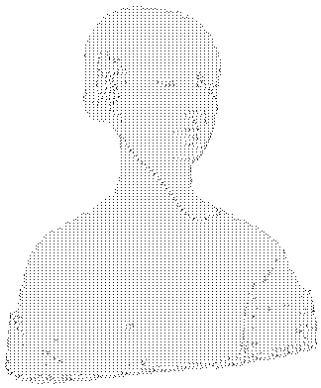


Figura 4.17: *I pixel salienti usati per la stima della scena del statua di Francesco Laurana.*

Figura 4.18: *L'ambiente di illuminazione stimato per la fotografia della statua del busto di donna.*

BSSRDF (ossia una Bidirectional Subsurface Scattering Reflectace Diffuse Function) la quale modella anche il comportamento della luce su vari livelli interni del materiale e viene tipicamente usata per simulare il comportamento di materiali come marmo e l'epidermide. Inoltre la statua è colorata, mentre un'assunzione fatta è quella di avere a che fare con oggetti colorati di bianco con luce bianca.

(a) Originale

(b) Reilluminato

Figura 4.19: *La statua con l'illuminazione dei neon davanti e dietro e la sua reilluminazione con l'ambiente stimato*

(a) Originale

(b) Reilluminato

Figura 4.20: *La statua con l'illuminazione naturale e la sua reilluminazione con l'ambiente stimato*

Capitolo 5

Conclusioni

Lo scopo di questa tesi è stato lo sviluppo di un sistema e di un metodo che consentisse di ricavare l'ambiente di illuminazione a partire da una fotografia di un oggetto e dal modello geometrico tridimensionale dell'oggetto stesso.

Poichè la luce è lineare, è possibile vedere la soluzione dell'equazione di rendering in maniera discreta come la soluzione di un sistema lineare del tipo $Ax = b$, dove A è una matrice che ha tante colonne quante sono le luci e tante righe quanti sono i pixel dell'immagine. In particolare la colonna j-esima rappresenta tutti i pixel illuminati dalla luce j-esima.

L'ambiente di illuminazione viene quindi visto come la combinazione lineare di un insieme di funzioni base: per queste la scelta è caduta sulle wavelet, poichè sono un supporto compatto per poter rappresentare una funzione, infatti già con poche basi è possibile avere una stima dell'ambiente di illuminazione con un errore abbstanza basso. Il problema è purtroppo molto malcondizionato, in quanto spesso si ritrovano colonne linearmente dipendenti, ad esempio se due luci molto simili per intensità e posizione illuminano l'oggetto, oppure righe linearmente dipendenti, quando i pixel sono molto vicini tra loro e quindi ricevono circa la stessa quantità di luce. Per questo nel calcolo del sistema lineare non vengono presi in considerazione tutti i pixel di tutte le immagini ma soltanto alcuni pixel, con un grado di significatività abbastanza differente.

Inoltre il processo di rendering è inerentemente lento poichè cerca di simulare in maniera corretta la fisica della luce calcolando l'equazione di rendering. Abbiamo scelto di utilizzare una soluzione più veloce anche se più approssimata per poter avere dei risultati nel più breve tempo possibile. Inoltre per poter velocizzare ulteriormente il calcolo, sono state usate altre approssimazioni come ad esempio il caching dell'ambient occlusion, la quale tiene conto dell'attenuazione della luce dovuta all'occlusione dell'oggetto ma aumenta il grado di correlazione dei punti illuminati e di conseguenza aumenta anche il malcondizionamento della matrice del sistema lineare risolto; la stima manuale della BRDF, che è differente tra i rendering delle basi e l'oggetto reale.

Nonostante questo si è potuto osservare un miglioramento rispetto ai risultati di Maschner [13], per i quali un'implementazione ha fornito un errore del 20% circa. In partcolare, il metodo differisce soprattutto per la scelta dei valori dei pixel da inserire nella matrice A. Marscher calcola il sistema $Ax = b$ considerando *tutti* i pixel dei rendering delle basi con cui suddivide l'ambiente di illuminazione: questo porta ad un malcondizionamento eccessivo, tanto da dover introdurre un termine (da impostare manualmente) che "mitighi" il malcondizionamento. Questo è dovuto al fatto che molti pixel di ogni rendering non sono illuminati e quindi contribuiscono soltanto a rendere instabile il sistema. Scegliendo soltanto un numero limitato e appropriato di pixel invece si diminuisce la dimensione del sistema e si riduce anche il malcondizionamento, sebbene non in maniera così elevata. Marschner inoltre non ha adottato una decomposizione tramite wavelet per rappresentare l'ambiente di illuminazione; questa consente, tramite pochi coefficienti, di poter stimare con una certa approssimazione un'ambiente di illuminazione.

Fortunatamente il progresso dell'hardware e degli algoritmi per il calcolo dell'equazione di rendering consentiranno di accelerare questa fase della stima, che è attualmente il collo di bottiglia del programma, rendendo

l'algoritmo scalabile.

L'output del programma è un ambiente di illuminazione, restituito sotto forma di cubical environment mapping. Esso può servire ad esempio per inserire in maniera realistica altri oggetti tridimensionali all'interno della fotografia, poichè è sufficiente illuminarli con l'ambiente stimato.

5.1 Sviluppi futuri

Uno delle possibili estensioni al lavoro proposto da questa tesi riguarda una differente scelta dei pixel salienti e delle basi che vanno a comporre i coefficienti della matrice per la risoluzione del sistema, ad esempio tramite metodi di *clustering*, raggruppando righe simili della matrice, oppure considerando il gradiente di un pixel per tutte le basi e non per un'unica base, come viene fatto attualmente tramite il filtro di Sobel. Un ulteriore approccio riguarderebbe l'implementazione e l'integrazione di questo lavoro con un framework teorico come quello proposto da Ramamoorti e al. [20].

Inoltre il lavoro svolto non tiene conto dell'albedo dell'oggetto considerato, in particolare questo viene sempre assunto bianco così come la luce.

Per poter stimare l'illuminazione inversa nel caso di oggetti colorati è necessario effettuare tre rendering differenti per ogni base, uno per ciascuna componente di colore R,G,B. Però per poter distinguere tra il colore e la luce (dato che nell'equazione di rendering questi due termini sono moltiplicati) è necessario uno shader (descritto in [10]) che proietti sul modello la texture map del colore dell'oggetto.

In questa maniera è possibile avere informazioni sull'albedo dell'oggetto (ottenuto, come spiegato in 2.4.4, dalla divisione dell'immagine proiettata e dell'illuminazione) in modo da poter distinguere ad esempio se un punto è in ombra oppure ha un colore scuro ma è illuminato.

Elenco delle figure

Elenco delle tabelle

Configurazione del programma

Il programma può essere configurato tramite il seguente file XML:

```
<LightRender waveletApproximation="2"
 waveletImageResolution="32"
 mesh="budda50k.ply"
 TexAlignXml="budda2.xml"
 photo="IMG_3854_piccola.exr"
 thresholdQuality="0.09"
 thresholdSobel="0.5"
 brdf="plastic"
 Kd="0.8" Ks="0.2" roughness="0.4"
 />
```

dove

- *waveletApproximation* è il livello di approssimazione j delle wavelet che si vuole raggiungere
- *waveletImageResolution* è la risoluzione dell'immagine
- *mesh* è il nome della mesh in formato .ply da caricare per effettuarne il render
- *TexAlignXml* è il file XML di TexAlign da cui vengono caricati i dati delle camera

- *photo* è la fotografia reale dalla quale vengono campionati i pixel salienti
- *thresholdQuality* è una soglia che indica quanta percentuale di pixel devono essere illuminati in una base perchè questa possa essere considerata valida
- *thresholdSobel* è una soglia per il filtro di Sobel
- *brdf* è il tipo di BRDF usata
- *Kd*, *Ks*, *roughness* sono i parametri della BRDF

Bibliografia

[1] Edward Angel. *Interactive Computer Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[2] James Arvo, Philip Dutré, Alexander Keller, Henrik Wann Jensen, Art Owen, Matt Pharr, and Peter Shirley. Monte carlo ray tracing. In *Siggraph 2003 Course Notes*.

[3] Rui Bastos, Kenneth Hoff, William Wynn, and Anselmo Lastra. Increased photorealism for interactive architectural walkthroughs. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 183–190, New York, NY, USA, 1999. ACM Press.

[4] Dario Bini, Milvio Capovani, and Ornella Menchi. *Metodi numerici per l'algebra lineare*. Zanichelli, 1988.

[5] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 189–198, New York, NY, USA, 1998. ACM Press.

[6] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. *Computer Graphics*, 31(Annual Conference Series):369–378, 1997.

[7] Kelly Dempski and Emmanuel Viale. *Advanced Lighting And Materials*. Wordware Publishing, Inc., 2004.

[8] Thomas Franken, Paolo Cignoni, Fabio Ganovelli, Matteo Dellepiane, Roberto Scopigno, and Claudio Montani. Assisting the user in image to geometry alignment. In *3D Digital Imaging and Modeling: Applications of Heritage, Industry, Medicine and Land, Workshop Italy-Canada*, 17-18 May 2005 2005.

[9] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[10] Larry Gritz. *Nvidia Gelato 2.0R5 Technical reference*.

[11] T. Inui, Y. Tanabe, and Y. Onodera. *Group theory and its applications in physics*. Springer Verlag, 1990.

[12] James T. Kajiya. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, New York, NY, USA, 1986. ACM Press.

[13] Stephen Robert Marschner. *Inverse Rendering For Computer Graphics*. PhD thesis, Cornell University, 1998.

[14] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 376–381, New York, NY, USA, 2003. ACM Press.

[15] M. Oren and S.K. Nayar. Generalization of Lambert's Reflectance Model. In *ACM 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 239–246, Jul 1994.

[16] Fabio Pellacini, Kiril Vidimce, Aaron Lefohn, Alex Mohr, Mark Leone, and John Warren. Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 464–470, New York, NY, USA, 2005. ACM Press.

[17] Matt Pharr and Greg Humphreys. *Physically Based Rendering*.

[18] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[19] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge (UK) and New York, 2nd edition, 1992.

[20] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 117–128. ACM Press / ACM SIGGRAPH, 2001.

[21] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(3):290–300, 2003.

[22] Peter Shirley. Time complexity for montecarlo radiosity. In *Eurographics '91*, pages 459–465. Elsevier Science Publishers, 1991.

[23] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. Wavelet for computer graphics: A primer. 1995.

[24] Chris Tchou, Jessi Stumpfel, Per Einarsson, Marcos Fajardo, and Paul Debevec. Estimating surface reflectance properties of a complex scene

under captured natural illumination. In *USC ICT Technical Report*, 2004.

[25] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *SIGGRAPH*, pages 265–272, 1992.

[26] Chris Wynn. An introduction to brdf-based lighting.

[27] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series*, pages 215–224, Los Angeles, California, USA, August 1999. Addison Wesley Longman.